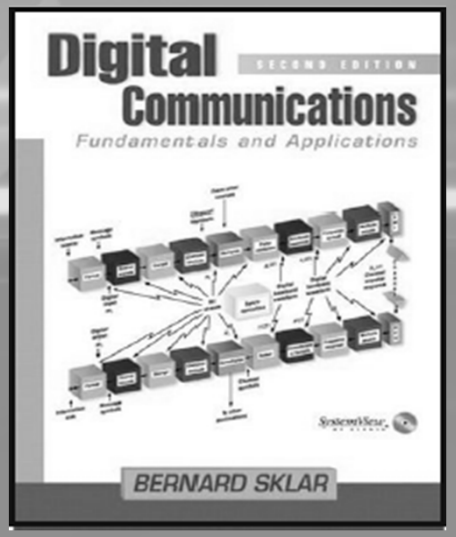


ENE 467
Digital Communications

TEACHING BY

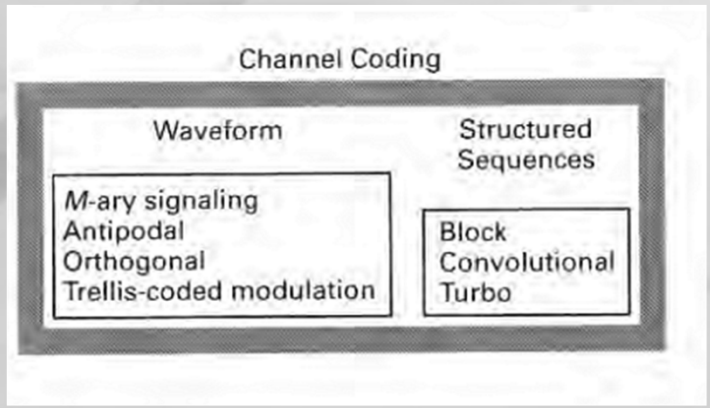
ASST. PROF. SUWAT PATTARAMALAI, PH.D.

6. Channel Coding: part I

- Outcome
  - Can explain waveform coding and structured sequences
  - Can explain types or Error control
  - Can explain Structured Sequences (models, code rate, parity code)
  - Can explain Linear Block Codes
  - Can explain Well-known block codes

Waveform Coding

Channel coding can be partitioned into two study areas, waveform (or signal design) coding and structured sequences (or structured redundancy), as shown in Figure 6.1. *Waveform coding* deals with transforming waveforms into "better waveforms," to make the detection process less subject to errors. *Structured sequences* deals with transforming data sequences into "better sequences," having structured redundancy (redundant bits). The redundant bits can then be used for the detection and correction of errors. The encoding procedure provides the coded signal (whether waveforms or structured sequences) with better distance properties than those of their uncoded counterparts.



- Antipodal Signals

Antipodal and orthogonal signals have been discussed earlier; we shall repeat the paramount features of these signal classes. The example shown in Figure 6.2 illustrates the analytical representation, $s_1(t) = -s_2(t) = \sin \omega_0 t$, $0 \le t \le T$, of a sinusoidal antipodal signal set, as well as its waveform representation and its vector representation. What are some synonyms or analogies that are used to describe *antipodal signals*? We can say that such signals are mirror images, or that one signal is the negative of the other, or that the signals are 180° apart.

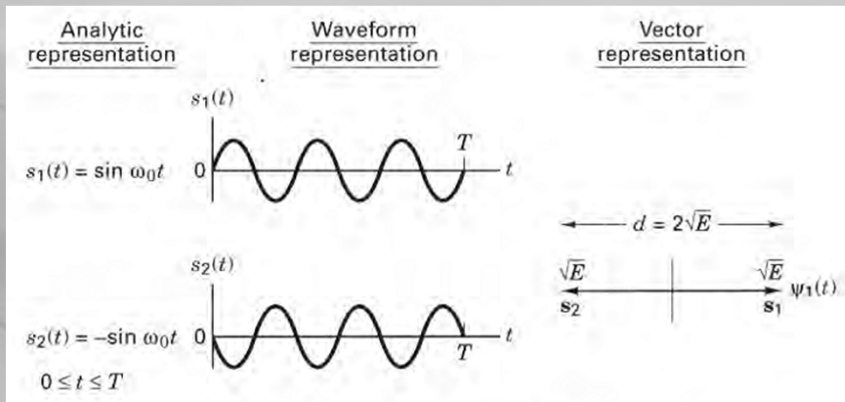


Figure 6.2 Example of an antipodal signal set.

Waveform Coding

- Orthogonal Signal

The example shown in Figure 6.3 illustrates an orthogonal signal set made up of pulse waveforms, described by

$$s_1(t) = p(t) \qquad 0 \le t \le T$$

and

$$s_2(t) = p\left(t - \frac{T}{2}\right) \qquad 0 \le t \le T$$

where $p(t)$ is a pulse with duration $\tau = T/2$, and T is the symbol duration. Another orthogonal waveform set frequently used in communication systems is sin x and cos x. In general, a set of equal-energy signals $s_i(t)$, where $i = 1, 2, \ldots, M$, constitutes an orthonormal (orthogonal, normalized to unity) set if and only if

$$z_{ij} = \frac{1}{E}\int_0^T s_i(t)s_j(t)\,dt = \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases} \qquad (6.1)$$

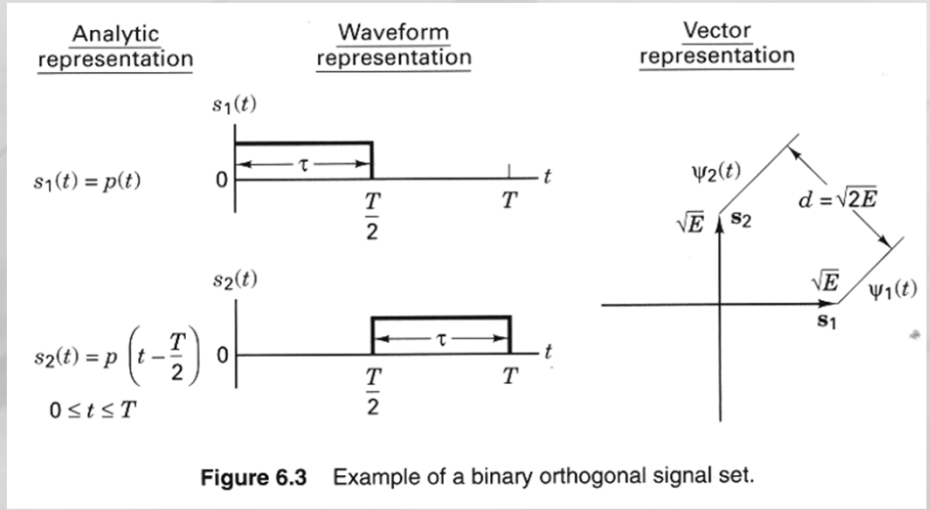


Figure 6.3 Example of a binary orthogonal signal set.

where z_{ij} is called the *cross-correlation coefficient*, and where E is the signal energy, expressed as

$$E = \int_0^T s_i^2(t)\,dt \qquad (6.2)$$

- M-ary Signaling

With M-ary signaling, the processor accepts k data bits at a time. It then instructs the modulator to produce one of $M = 2^k$ waveforms; binary signaling is the special case where $k = 1$. For $k > 1$, M-ary signaling alone can be regarded as a *waveform coding* procedure. For orthogonal signaling (e.g., MFSK), as k increases there is improved error performance or a reduction in required E_b/N_0, at the expense of bandwidth; nonorthogonal signaling (e.g., MPSK) manifests improved bandwidth efficiency, at the expense of degraded error performance or an increase in required E_b/N_0. By the appropriate choice of signal waveforms, one can trade off error probability, E_b/N_0, and bandwidth efficiency.

Waveform Coding

- Orthogonal Codes

The orthogonality condition in Equation (6.1) is presented in terms of waveforms $s_i(t)$ and $s_j(t)$, where $i, j = 1, \ldots, M$, and M is the size of the waveform set. Each waveform in the set $\{s_i(t)\}$ may consist of a sequence of pulses, where each pulse is designated with a level +1 or −1, which in turn represents the binary digit 1 or 0, respectively. When the set is expressed in this way, Equation (6.1) can be simplified by stating that $\{s_i(t)\}$ constitutes an orthogonal set if and only if

$$z_{ij} = \frac{\text{number of digit agreements} - \text{number of digit disagreements}}{\text{total number of digits in the sequence}} \qquad (6.3)$$

$$= \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases}$$

A one-bit data set can be transformed, using *orthogonal codewords* of two digits each, described by the rows of matrix $\mathbf{H}_1$ as follows:

Data set	Orthogonal codeword set
0	$\mathbf{H}_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}$ (6.4a)
1	

For this, and the following examples, use Equation (6.3) to verify the orthogonality of the codeword set. To encode a 2-bit data set, we extend the foregoing set both horizontally and vertically, creating matrix $\mathbf{H}_2$.

Data set	Orthogonal codeword set
0 0	$\mathbf{H}_2 = \left[\begin{array}{cc:cc} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ \hdashline 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{array}\right] = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_1 \\ \mathbf{H}_1 & \overline{\mathbf{H}_1} \end{bmatrix}$ (6.4b)
0 1	
1 0	
1 1	

The lower right quadrant is the complement of the prior codeword set. We continue the same construction rule to obtain an orthogonal set $\mathbf{H}_3$ for a 3-bit data set.

Data Set	Orthogonal codeword set
0 0 0	$\mathbf{H}_3 = \left[\begin{array}{cccc:cccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ \hdashline 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] = \begin{bmatrix} \mathbf{H}_2 & \mathbf{H}_2 \\ \mathbf{H}_2 & \overline{\mathbf{H}_2} \end{bmatrix}$ (6.4c)
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	

In general, we can construct a codeword set $\mathbf{H}_k$, of dimension $2^k \times 2^k$, called a *Hadamard matrix,* for a k-bit data set from the $\mathbf{H}_{k-1}$ matrix, as follows:

$$\mathbf{H}_k = \begin{bmatrix} \mathbf{H}_{k-1} & \mathbf{H}_{k-1} \\ \mathbf{H}_{k-1} & \overline{\mathbf{H}_{k-1}} \end{bmatrix} \qquad (6.4d)$$

Each pair of words in each codeword set $\mathbf{H}_1, \mathbf{H}_2, \mathbf{H}_3, \ldots, \mathbf{H}_k, \ldots$ has as many digit agreements as disagreements [2]. Hence, in accordance with Equation (6.3), $z_{ij} = 0$ (for $i \neq j$), and each of the sets is orthogonal.

Waveform Coding

- Biorthogonal Codes

The orthogonality condition in Equation (6.1) is presented in terms of waveforms $s_i(t)$ and $s_j(t)$, where $i, j = 1, \ldots, M$, and M is the size of the waveform set. Each waveform in the set $\{s_i(t)\}$ may consist of a sequence of pulses, where each pulse is designated with a level +1 or −1, which in turn represents the binary digit 1 or 0, respectively. When the set is expressed in this way, Equation (6.1) can be simplified by stating that $\{s_i(t)\}$ constitutes an orthogonal set if and only if

$$z_{ij} = \frac{\text{number of digit agreements} - \text{number of digit disagreements}}{\text{total number of digits in the sequence}} \qquad (6.3)$$

$$= \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases}$$

A *biorthogonal* signal set of M total signals or codewords can be obtained from an orthogonal set of $M/2$ signals by augmenting it with the negative of each signal as follows:

$$\mathbf{B}_k = \begin{bmatrix} \mathbf{H}_{k-1} \\ \hline -\mathbf{H}_{k-1} \end{bmatrix}$$

For example, a 3-bit data set can be transformed into a biothogonal codeword set as follows:

Data set	Biorthogonal codeword set
0 0 0	0 0 0 0
0 0 1	0 1 0 1
0 1 0	0 0 1 1
0 1 1	0 1 1 0
	$\mathbf{B}_3 =$ ----------
1 0 0	1 1 1 1
1 0 1	1 0 1 0
1 1 0	1 1 0 0
1 1 1	1 0 0 1

The biorthogonal set is really two sets of orthogonal codes such that each codeword in one set has its antipodal codeword in the other set. The biorthogonal set consists of a *combination of orthogonal and antipodal* signals. With respect to z_{ij} of Equation (6.1), biorthogonal codes can be characterized as

$$z_{ij} = \begin{cases} 1 & \text{for } i = j \\ -1 & \text{for } i \neq j,\ |i - j| = \dfrac{M}{2} \\ 0 & \text{for } i \neq j,\ |i - j| \neq \dfrac{M}{2} \end{cases} \qquad (6.8)$$

One advantage of a biorthogonal code over an orthogonal one for the same data set is that the biorthogonal code requires *one-half* as many code bits per codeword (compare the rows of the $\mathbf{B}_3$ matrix with those of the $\mathbf{H}_3$ matrix presented earlier). Thus the bandwidth requirements for biorthogonal codes are one-half the requirements for comparable orthogonal ones. Since antipodal signal vectors have better distance properties than orthogonal ones, it should come as no surprise that biorthogonal codes perform slightly better than orthogonal ones. For equally likely, equal-energy biorthogonal signals, the probability of codeword (symbol) error can be upper bounded, as follows [2]:

Waveform Coding

- Transorthogonal (Simplex) Codes

The orthogonality condition in Equation (6.1) is presented in terms of waveforms $s_i(t)$ and $s_j(t)$, where $i, j = 1, \ldots, M$, and M is the size of the waveform set. Each waveform in the set $\{s_i(t)\}$ may consist of a sequence of pulses, where each pulse is designated with a level +1 or −1, which in turn represents the binary digit 1 or 0, respectively. When the set is expressed in this way, Equation (6.1) can be simplified by stating that $\{s_i(t)\}$ constitutes an orthogonal set if and only if

$$z_{ij} = \frac{\text{number of digit agreements} - \text{number of digit disagreements}}{\text{total number of digits in the sequence}} \tag{6.3}$$

$$= \begin{cases} 1 & \text{for } i = j \\ 0 & \text{otherwise} \end{cases}$$

A code generated from an orthogonal set by deleting the first digit of each codeword is called a *transorthogonal* or *simplex code*. Such a code is characterized by

$$z_{ij} = \begin{cases} 1 & \text{for } i = j \\ \dfrac{-1}{M-1} & \text{for } i \neq j \end{cases} \tag{6.11}$$

A simplex code represents the *minimum energy* equivalent (in the error-probability sense) of the equally likely orthogonal set. In comparing the error performance of orthogonal, biorthogonal, and simplex codes, we can state that simplex coding requires the minimum E_b/N_0 for a specified symbol error rate. However, for a *large value of M*, all three schemes are *essentially identical* in error performance. Biorthogonal coding requires half the bandwidth of the others. But for each of these codes, bandwidth requirements (and system complexity) grow exponentially with the value of M; therefore, such coding schemes are attractive only when large bandwidths are available.

Waveform Coding

- Waveform Coding System Example

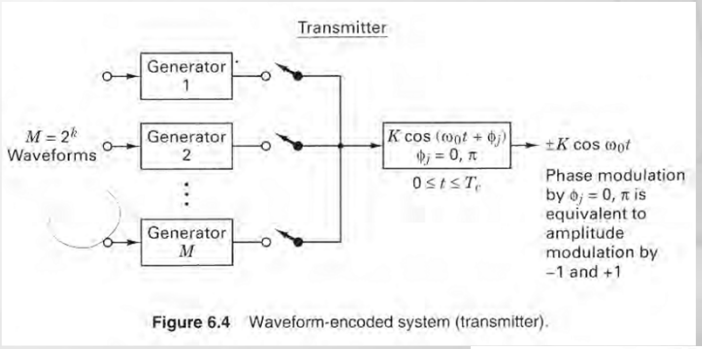


Figure 6.4 Waveform-encoded system (transmitter).

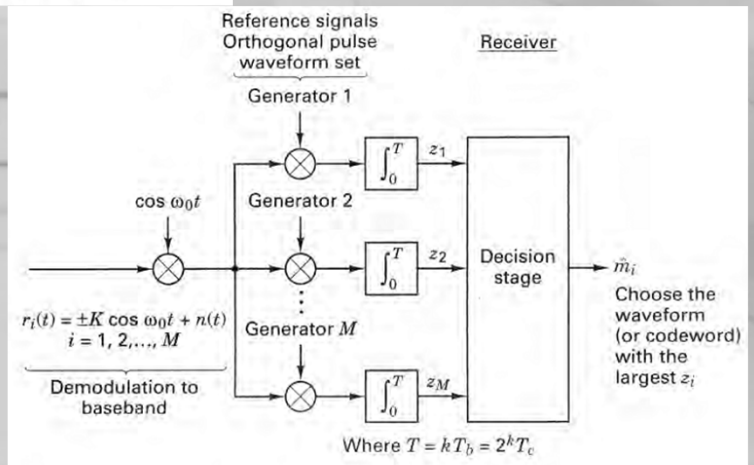


Figure 6.5 Waveform-encoded system with coherent detection (receiver).

Figure 6.4 illustrates an example of assigning a k-bit message from a message set of size $M = 2^k$, with a coded-pulse sequence from a code set of the same size. Each k-bit message chooses one of the generators yielding a coded-pulse sequence or codeword. The sequences in the coded set that replace the messages form a waveform set with good distance properties (e.g., orthogonal, biorthogonal). For the orthogonal code described in Section 6.1.3.1, each codeword consists of $M = 2^k$ pulses (representing code bits). Hence 2^k code bits replace k message bits. The chosen sequence then modulates a carrier wave using binary PSK, such that the phase ($\phi_i = 0$ or π) of the carrier during each code-bit duration, $0 \le t \le T_c$, corresponds to the amplitude ($j = -1$ or 1) of the jth bipolar pulse in the codeword. At the receiver in Figure 6.5, the signal is demodulated to baseband and fed to M correlators (or matched filters). For orthogonal codes, such as those characterized by the Hadamard matrix in Section 6.1.3.1, correlation is performed over a codeword duration that can be expressed as $T = 2^k T_c$. For a real-time communication system, messages may not be delayed; hence, the codeword duration must be the same as the message duration, and thus, T can also be expressed as $T = (\log_2 M)\, T_b = kT_b$, where T_b is the message-bit duration. Note that the time duration of a message bit is M/k times longer than that of a code bit. In other words, the code bits or coded pulses (which are PSK modulated) must move at a rate M/k faster than the message bits. For such orthogonally coded waveforms and an AWGN channel, the expected value at the output of each correlator, at time T, is zero, except for the correlator corresponding to the transmitted codeword.

What is the advantage of such orthogonal waveform coding compared with simply sending one bit or one pulse at a time? One can compare the bit-error performance with and without such coding by comparing Equation (4.79) for coherent detection of antipodal signals with Equation (6.7) for the coherent detection of orthogonal codewords. For a given size k-bit message (say, $k = 5$) and a desired bit-error probability (say, 10^{-5}), the detection of orthogonal codewords (each having a 5-bit meaning) can be accomplished with about 2.9 dB less E_b/N_0 than the bit-by-bit detection of antipodal signals. (The demonstration is left as an exercise for the reader in Problem 6.28.) One might have guessed this result by comparing the performance curves for orthogonal signaling in Figure 4.28 with the binary (antipodal) curve in Figure 4.29. What price do we pay for this error-performance improvement? The cost is more transmission bandwidth. In this example, transmission of an uncoded message consists of sending 5 bits. With coding, how many coded pulses must be transmitted for each message sequence? With the waveform coding of this example, each 5-bit message sequence is represented by $M = 2^k = 2^5 = 32$ code bits or coded pulses. The 32 coded pulses in a codeword must be sent in the same time duration as the corresponding 5 bits from which they stem. Thus, the required transmission bandwidth is 32/5 times that of the uncoded case. In general, the bandwidth needed for such orthogonally coded signals is M/k times greater than that needed for the uncoded case. Later, more efficient ways to trade off the benefits of coding versus bandwidth [3, 4] will be examined.

Types of Error Control

- Terminal Connectivity

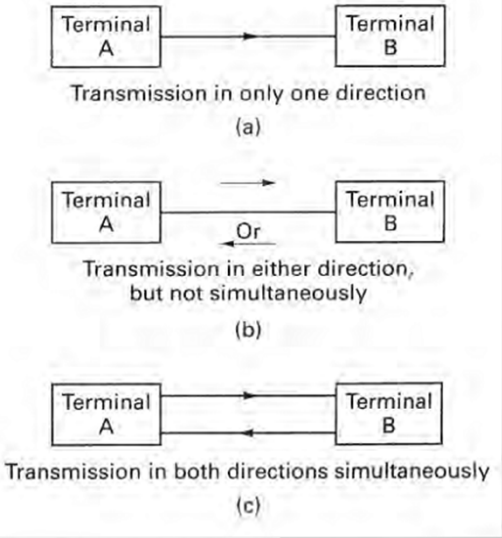
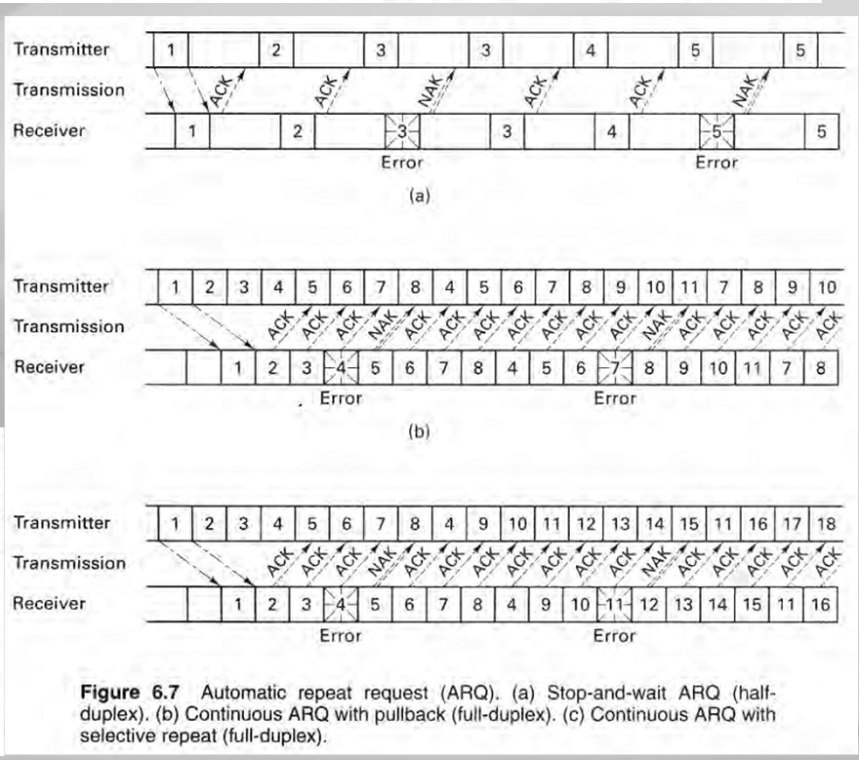


Figure 6.6 Terminal connectivity classifications. (a) Simplex. (b) Half-duplex. (c) Full-duplex.

Before we discuss the details of structured redundancy, let us describe the two basic ways such redundancy is used for controlling errors. The first, *error detection and retransmission,* utilizes *parity bits* (redundant bits added to the data) to detect that an error has been made. The receiving terminal does not attempt to correct the error; it simply requests that the transmitter retransmit the data. Notice that a two-way link is required for such dialogue between the transmitter and receiver. The second type of error control, *forward error correction* (FEC), requires a one-way link only, since in this case the parity bits are designed for both the detection and correction of errors. We shall see that not all error patterns can be corrected; error-correcting codes are classified according to their error-correcting capabilities.

Automatic Repeat Request



The major advantage of ARQ over forward error correction (FEC) is that error detection requires much simpler decoding equipment and much less redundancy than does error correction. Also, ARQ is adaptive in the sense that information is retransmitted only when errors occur. On the other hand, FEC may be desirable in place of, or in addition to, error detection, for any of the following reasons:

1. A reverse channel is not available or the delay with ARQ would be excessive.
2. The retransmission strategy is not conveniently implemented.
3. The expected number of errors, without corrections, would require excessive retransmissions.

Figure 6.7 Automatic repeat request (ARQ). (a) Stop-and-wait ARQ (half-duplex). (b) Continuous ARQ with pullback (full-duplex). (c) Continuous ARQ with selective repeat (full-duplex).

Structured Sequences

- Channel Models

6.3.1.1 Discrete Memoryless Channel

A *discrete memoryless channel* (DMC) is characterized by a discrete input alphabet, a discrete output alphabet, and a set of conditional probabilities $P(j|i)$ $(1 \le i \le M, 1 \le j \le Q)$, where i represents a modulator M-ary input symbol, j represents a demodulator Q-ary output symbol, and $P(j|i)$ is the probability of receiving j given that i was transmitted. Each output symbol of the channel depends only on the corresponding input, so that for a given input sequence $\mathbf{U} = u_1, u_2, \ldots, u_m, \ldots, u_N$, the conditional probability of a corresponding output sequence $\mathbf{Z} = z_1, z_2, \ldots, z_m, \ldots, z_N$ may be expressed as

$$P(\mathbf{Z}|\mathbf{U}) = \prod_{m=1}^{N} P(z_m|u_m) \tag{6.12}$$

In the event that the channel *has memory* (i.e., noise or fading that occurs in bursts), the conditional probability of the sequence $\mathbf{Z}$ would need to be expressed as the *joint* probability of all the elements of the sequence. Equation (6.12) expresses the *memoryless* condition of the channel. Since the channel noise in a memoryless channel is defined to affect each symbol independently of all the other symbols, the conditional probability of $\mathbf{Z}$ is seen as the product of the independent element probabilities.

Structured Sequences

- Channel Models

6.3.1.2 Binary Symmetric Channel

A *binary symmetric channel* (BSC) is a special case of a DMC; the input and output alphabet sets consist of the binary elements (0 and 1). The conditional probabilities are symmetric:

$$P(0|1) = P(1|0) = p$$

and

$$P(1|1) = P(0|0) = 1 - p \tag{6.13}$$

Equation (6.13) expresses the channel *transition probabilities*. That is, given that a channel symbol was transmitted, the probability that it is received in error is p (related to the symbol energy), and the probability that it is received correctly is $(1-p)$. Since the demodulator output consists of the discrete elements 0 and 1, the demodulator is said to make a firm or *hard decision* on each symbol. A commonly used code system consists of BPSK modulated coded data and hard decision demodulation. Then the channel symbol error probability is found using the methods discussed in Section 4.7.1 and Equation (4.79) to be

$$p = Q\left(\sqrt{\frac{2E_c}{N_0}}\right)$$

where E_c/N_0 is the channel symbol energy per noise density, and $Q(x)$ is defined in Equation (3.43).

When such hard decisions are used in a binary coded system, the demodulator feeds the two-valued *code symbols* or *channel bits* to the decoder. Since the decoder then operates on the hard decisions made by the demodulator, decoding with a BSC channel is called *hard-decision decoding*.

Structured Sequences

- Channel Models

It is possible to design decoders using soft decisions, but block code soft-decision decoders are substantially more complex than hard-decision decoders; therefore, block codes are usually implemented with hard-decision decoders. For convolutional codes, both hard- and soft-decision implementations are equally popular. In this chapter we consider that the channel is a binary symmetric channel (BSC), and hence the decoder employs hard decisions. In Chapter 7 we further discuss channel models, as well as hard- versus soft-decision decoding for convolutional codes.

6.3.1.3 Gaussian Channel

We can generalize our definition of the DMC to channels with alphabets that are not discrete. An example is the *Gaussian channel* with a discrete input alphabet and a continuous output alphabet over the range $(-\infty, \infty)$. The channel adds noise to the symbols. Since the noise is a Gaussian random variable with zero mean and variance σ^2, the resulting probability density function (pdf) of the received random variable z, conditioned on the symbol u_k (the likelihood of u_k), can be written as

$$p(z|u_k) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left[\frac{-(z-u_k)^2}{2\sigma^2}\right] \tag{6.14}$$

for all z, where $k = 1, 2, \ldots, M$. For this case, *memoryless* has the same meaning as it does in Section 6.3.1.1, and Equation (6.12) can be used to obtain the conditional probability for the sequence $\mathbf{Z}$.

When the demodulator output consists of a continuous alphabet or its quantized approximation (with greater than two quantization levels), the demodulator is said to make *soft decisions*. In the case of a coded system, the demodulator feeds such quantized code symbols to the decoder. Since the decoder then operates on the soft decisions made by the demodulator, decoding with a Gaussian channel is called *soft-decision decoding*.

In the case of a hard-decision channel, we are able to characterize the detection process with a channel symbol error probability. However, in the case of a soft-decision channel, the detector makes the kind of decisions (soft decisions) that cannot be labeled as correct or incorrect. Thus, since there are no firm decisions, there cannot be a probability of making an error; the detector can only formulate a family of conditional probabilities or likelihoods of the different symbol types.

Structured Sequences

6.3.2 Code Rate and Redundancy

In the case of block codes, the source data are segmented into blocks of k data bits, also called information bits or message bits; each block can represent any one of 2^k distinct messages. The encoder transforms each k-bit data block into a larger block of n bits, called code bits or channel symbols. The $(n - k)$ bits, which the encoder adds to each data block, are called *redundant bits, parity bits,* or *check bits;* they carry no new information. The code is referred to as an (n, k) code. The ratio of redundant bits to data bits, denoted $(n - k)/k$, within a block is called the *redundancy* of the code; the ratio of data bits to total bits, k/n, is called the *code rate.* The code rate can be thought of as the portion of a code bit that constitutes information. For example, in a rate $\frac{1}{2}$ code, each code bit carries $\frac{1}{2}$ bit of information.

In this chapter and in Chapters 7 and 8 we consider those coding techniques that provide redundancy by increasing the required transmission bandwidth. For example, an error control technique that employs a rate 1/2 code (100% redundancy) will require double the bandwidth of an uncoded system. However, if a rate 3/4 code is used, the redundancy is 33% and the bandwidth expansion is only 4/3. In Chapter 9 we consider modulation/coding techniques for bandlimited channels where complexity instead of bandwidth is traded for error performance improvement.

6.3.2.1 Code-Element Nomenclature

Different authors describe an encoder's output elements in a variety of ways: code bits, channel bits, code symbols, channel symbols, parity bits, parity symbols. The terms are all very similar. In this text, for a binary code, the terms "code bit," "channel bit," "code symbol," and "channel symbol" have exactly the same meaning. The terms "code bit" and "channel bit" are most descriptive for binary codes only. The more generic names "code symbol" and "channel symbol" are often preferred because they can be used to describe binary or nonbinary codes equally well. Note that such code symbols or channel symbols are not to be confused with the grouping of bits to form transmission symbols that was done in previous chapters. The terms "parity bit" and "parity symbol" are used to identify only those code elements that represent the redundancy components added to the original data.

Structured Sequences

6.3.3 Parity-Check Codes

6.3.3.1 Single-Parity-Check Code

Parity-check codes use linear sums of the information bits, called *parity symbols* or *parity bits,* for error detection or correction. A single-parity-check code is constructed by adding a single-parity bit to a block of data bits. The parity bit takes on the value of one or zero as needed to ensure that the summation of all the bits in the codeword yields an even (or odd) result. The summation operation is performed using modulo-2 arithmetic (exclusive-or logic), as described in Section 2.9.3. If the added parity is designed to yield an even result, the method is termed *even parity;* if it is designed to yield an odd result, the method is termed *odd parity.* Figure 6.8a illustrates a serial data transmission (the rightmost bit is the earliest bit). A single-parity bit is added (the leftmost bit in each block) to yield even parity.

At the receiving terminal, the decoding procedure consists of testing that the modulo-2 sum of the codeword bits yields a zero result (even parity). If the result is found to be one instead of zero, the codeword is known to contain errors. The rate of the code can be expressed as $k/(k+1)$. Do you suppose the decoder can automatically *correct* a digit that is received in error? No, it cannot. It can only *detect* the presence of an odd number of bit errors. (If an even number of bits are inverted, the parity test will appear correct, which represents the case of an *undetected error.*) Assuming that

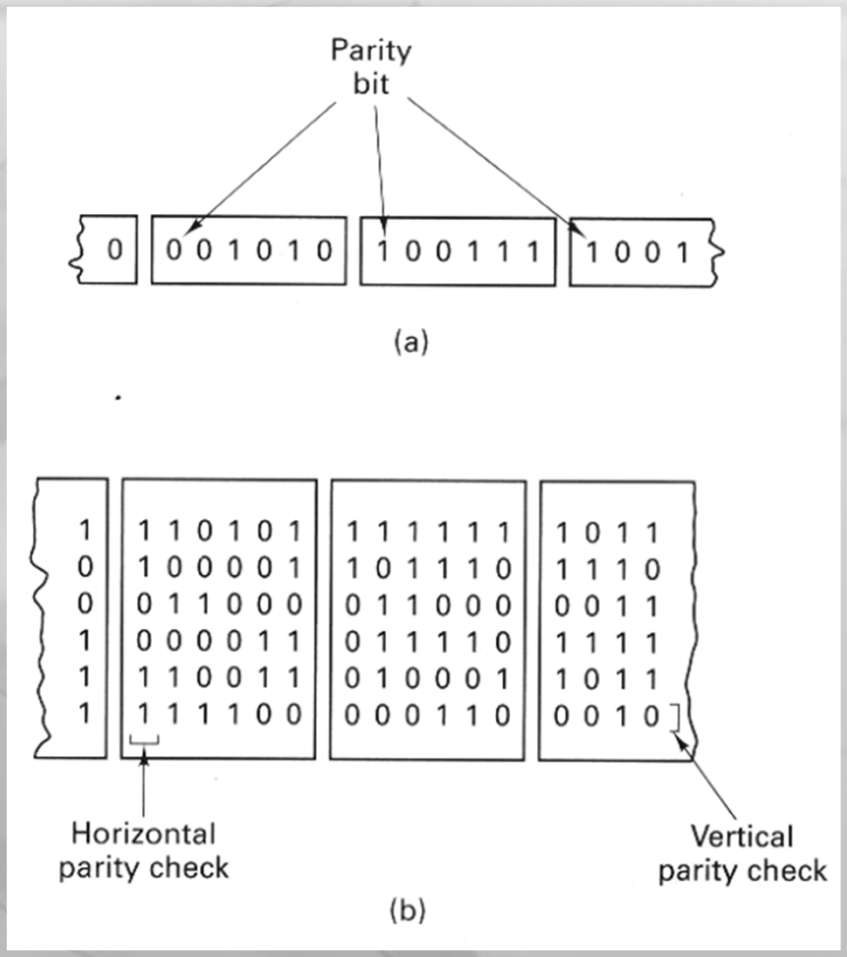


Figure 6.8 Parity checks for serial and parallel code structures. (a) Serial structure. (b) Parallel structure.

Structured Sequences

all bit errors are equally likely and occur independently, we can write the probability of j errors occurring in a block of n symbols as

$$P(j, n) = \binom{n}{j} p^j (1 - p)^{n-j} \tag{6.15}$$

where p is the probability that a *channel symbol* is received in error, and where

$$\binom{n}{j} = \frac{n!}{j!(n - j)!} \tag{6.16}$$

is the number of various ways in which j bits out of n may be in error. Thus, for a single-parity error-detection code, the probability of an undetected error P_{nd} with a block of n bits is computed as follows:

$$P_{nd} = \sum_{j=1}^{\substack{n/2 \text{ (for } n \text{ even)} \\ (n-1)/2 \text{ (for } n \text{ odd)}}} \binom{n}{2j} p^{2j} (1 - p)^{n-2j} \tag{6.17}$$

Example 6.1 Even-Parity Code

Configure a (4, 3) even-parity error-detection code such that the parity symbol appears as the leftmost symbol of the codeword. Which error patterns can the code detect? Compute the probability of an undetected message error, assuming that all symbol errors are independent events and that the probability of a channel symbol error is $p = 10^{-3}$.

Solution

Message	Parity	Codeword
000	0	0 000
100	1	1 100
010	1	1 010
110	0	0 110
001	1	1 001
101	0	0 101
011	0	0 011
111	1	1 111

parity message

The code is capable of detecting all single- and triple-error patterns. The probability of an undetected error is equal to the probability that two or four errors occur anywhere in a codeword.

$$\begin{aligned} P_{nd} &= \binom{4}{2} p^2 (1 - p)^2 + \binom{4}{4} p^4 \\ &= 6p^2(1 - p)^2 + p^4 \\ &= 6p^2 - 12p^3 + 7p^4 \\ &= 6(10^{-3})^2 - 12(10^{-3})^3 + 7(10^{-3})^4 \approx 6 \times 10^{-6} \end{aligned}$$

Structured Sequences

6.3.3.2 Rectangular Code

A *rectangular code,* also called a *product code,* can be thought of as a parallel code structure, depicted in Figure 6.8b. First we form a rectangle of message bits comprising M rows and N columns; then, a horizontal parity check is appended to each row and a vertical parity check is appended to each column, resulting in an augmented array of dimensions $(M + 1) \times (N + 1)$. The rate of the rectangular code k/n can then be written as

$$\frac{k}{n} = \frac{MN}{(M + 1)(N + 1)}$$

How much more powerful is the rectangular code than the single-parity code, which is only capable of error detection? Notice that any single bit error will cause a parity check failure in one of the array columns *and* in one of the array rows. Therefore, the rectangular code can correct any single error pattern since such an error is uniquely located at the intersection of the error-detecting row and the error-detecting column. For the example shown in Figure 6.8b, the array dimensions are $M = N = 5$; therefore, the figure depicts a (36, 25) code that can correct a single error located anywhere in the 36-bit positions. For such an error-correcting block code, we compute the probability that the decoded block has an uncorrected error by accounting for all the ways in which a *message error* can be made. Starting with the probability of j errors in a block of n symbols, expressed in Equation (6.15), we can write the probability of a message error, also called a *block error* or *word error,* for a code that can correct all t and fewer error patterns as

$$P_M = \sum_{j=t+1}^{n} \binom{n}{j} p^j (1 - p)^{n-j} \tag{6.18}$$

where p is the probability that a *channel symbol* is received in error. For the example in Figure 6.8b, the code can correct all single error patterns ($t = 1$) within the rectangular block of $n = 36$ bits. Hence, the summation in Equation (6.18) starts with $j = 2$:

$$P_M = \sum_{j=2}^{36} \binom{36}{j} p^j (1 - p)^{36-j}$$

When p is reasonably small, the first term in the summation is the dominant one; Therefore, for this (36, 25) rectangular code example, we can write

$$P_M \approx \binom{36}{2} p^2 (1 - p)^{34}$$

The *bit error probability* P_B depends on the particular code and decoder. An approximation for P_B is given in Section 6.5.3.

Structured Sequences

6.3.4 Why Use Error-Correction Coding?

Error-correction coding can be regarded as a vehicle for effecting various system trade-offs. Figure 6.9 compares two curves depicting bit-error performance versus

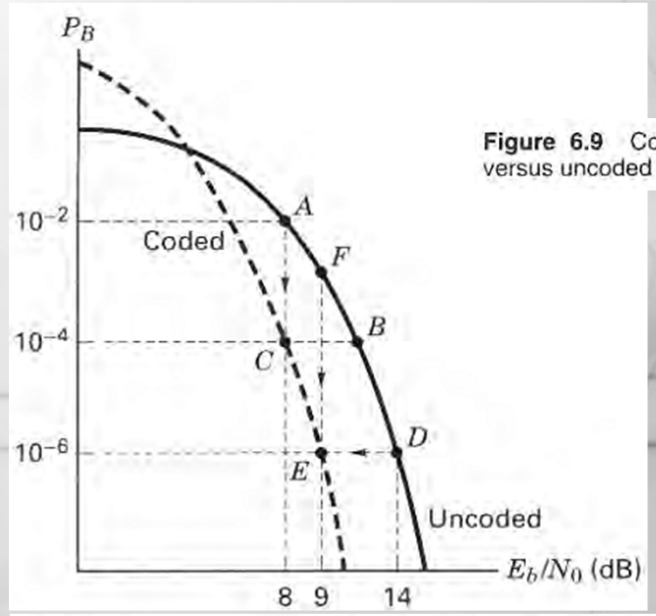


Figure 6.9 Comparison of typical coded versus uncoded error performance.

E_b/N_0. One curve represents a typical modulation scheme without coding. The second curve represents the same modulation with coding. Demonstrated below are four benefits or trade-offs that can be achieved with the use of channel coding.

6.3.4.1 Trade-Off 1: Error Performance versus Bandwidth

Imagine that a simple, inexpensive voice communication system has just been developed and delivered to a customer. The system does not use error-correction coding. Consider that the operating point of the system can be depicted by point A in Figure 6.9 ($E_b/N_0 = 8$ dB, and $P_B = 10^{-2}$). After a few trials, there are complaints about the voice quality; the customer suggests that the bit-error probability should be lowered to 10^{-4}. The usual way of obtaining better error performance in such a system would be by effecting an operating point movement from point A to, say, point B in Figure 6.9. However, suppose that the E_b/N_0 of 8 dB is the most that is available in this system. The figure suggests that one possible trade-off is to move the operating point from point A to point C. That is, "walking" down the vertical line to point C on the coded curve can provide the customer with improved error performance. What does it cost? Aside from the new components (encoder and decoder) needed, the price is more transmission bandwidth. Error-correction coding needs redundancy. If we assume that the system is a real-time communication system (such that the message may not be delayed), the addition of redundant bits dictates a faster rate of transmission, which of course means more bandwidth.

6.3.4.2 Trade-Off 2: Power versus Bandwidth

Consider that a system without coding, operating at point D in Figure 6.9 ($E_b/N_0 = 14$ dB, and $P_B = 10^{-6}$), has been delivered to a customer. The customer has no complaints about the quality of the data, but the equipment is having some reliability problems as a result of providing an E_b/N_0 of 14 dB. In other words, the equipment keeps breaking down. If the requirement on E_b/N_0 or power could be reduced, the reliability difficulties might also be reduced. Figure 6.9 suggests a trade-off by moving the operating point from point D to point E. That is, if error-correction coding is introduced, a reduction in the required E_b/N_0 can be achieved. Thus, the trade-off is one in which the same quality of data is achieved, but the coding allows for a reduction in power or E_b/N_0. What is the cost? The same as before—more bandwidth.

Notice that for *non-real-time* communication systems, error-correction coding can be used with a somewhat different trade-off. It is possible to obtain improved bit-error probability or reduced power (similar to trade-off 1 or 2 above) by paying the price of *delay* instead of bandwidth.

Structured Sequences

6.3.4 Why Use Error-Correction Coding?

Error-correction coding can be regarded as a vehicle for effecting various system trade-offs. Figure 6.9 compares two curves depicting bit-error performance versus

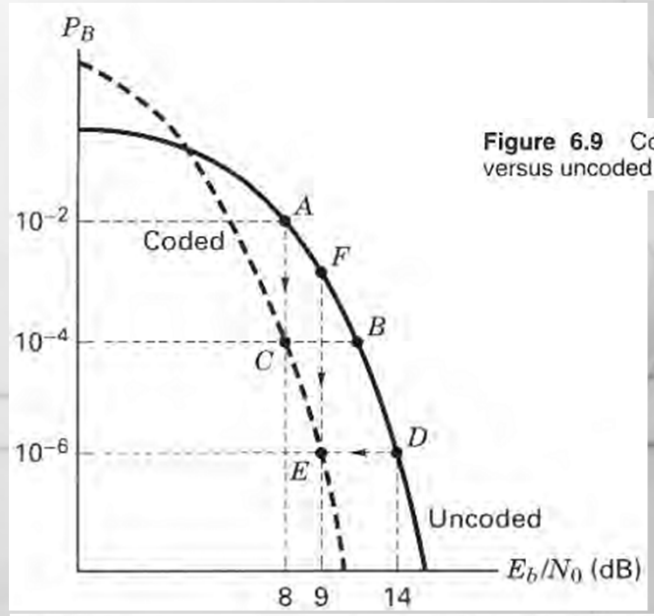


Figure 6.9 Comparison of typical coded versus uncoded error performance.

E_b/N_0. One curve represents a typical modulation scheme without coding. The second curve represents the same modulation with coding. Demonstrated below are four benefits or trade-offs that can be achieved with the use of channel coding.

6.3.4.3 Coding Gain

The trade-off example described in the previous section has allowed a reduction in E_b/N_0 from 14 dB to 9 dB, while maintaining the same error performance. In the context of this example and Figure 6.9, we now define *coding gain*. For a *given bit-error probability*, coding gain is defined as the "relief" or reduction in E_b/N_0 that can be realized through the use of the code. Coding gain G is generally expressed in dB, such as

$$G(\text{dB}) = \left(\frac{E_b}{N_0}\right)_u (\text{dB}) - \left(\frac{E_b}{N_0}\right)_c (\text{dB}) \tag{6.19}$$

where $(E_b/N_0)_u$ and $(E_b/N_0)_c$ represent the required E_b/N_0, uncoded and coded, respectively.

6.3.4.4 Trade-Off 3: Data Rate versus Bandwidth

Consider that a system without coding, operating at point D in Figure 6.9 ($E_b/N_0 = 14$ dB, and $P_B = 10^{-6}$) has been developed. Assume that there is no problem with the data quality and no particular need to reduce power. However, in this example, suppose that the customer's data rate requirement increases. Recall the relationship in Equation (5.20b):

$$\frac{E_b}{N_0} = \frac{P_r}{N_0}\left(\frac{1}{R}\right)$$

If we do nothing to the system except increase the data rate R, the above expression shows that the received E_b/N_0 would decrease, and in Figure 6.9, the operating point would move upwards from point D to, let us say, some point F. Now, envision "walking" down the vertical line to point E on the curve that represents coded modulation. Increasing the data rate has degraded the quality of the data. But, the use of error-correction coding brings back the same quality at the same power level (P_r/N_0). The E_b/N_0 is reduced, but the code facilitates getting the same error probability with a lower E_b/N_0. What price do we pay for getting this higher data rate or greater capacity? The same as before—increased bandwidth.

Structured Sequences

6.3.4.5 Trade-Off 4: Capacity versus Bandwidth

Trade-off 4 is similar to trade-off 3 because both achieve increased capacity. A spread-spectrum multiple access technique, called code-division multiple access (CDMA) and described in Chapter 12, is one of the schemes used in cellular telephony. In CDMA, where users simultaneously share the same spectrum, each user is an interferer to each of the other users in the same cell or nearby cells. Hence, the capacity (maximum number of users) per cell is inversely proportional to E_b/N_0. (See Section 12.8.) In this application, a lowered E_b/N_0 results in a raised capacity; the code achieves a reduction in each user's power, which in turn allows for an increase in the number of users. Again, the cost is more bandwidth. But, in this case, the signal-bandwidth expansion due to the error-correcting code is small compared with the more significant spread-spectrum bandwidth expansion, and thus, there is no impact on the transmission bandwidth.

In each of the above trade-off examples, a "traditional" code involving redundant bits and faster signaling (for a real-time communication system) has been assumed; hence, in each case, the cost was expanded bandwidth. However, there exists an error-correcting technique, called *trellis-coded modulation*, that does not require faster signaling or expanded bandwidth for real-time systems. (This technique is described in Section 9.10.)

6.3.4.6 Code Performance at Low Values of E_b/N_0

The reader is urged to solve Problem 6.5, which is similar to Example 6.2. In part (a) of Problem 6.5, where an E_b/N_0 of 14 dB is given, the result is a message-error performance improvement through the use of coding. However, in part (b) where the E_b/N_0 has been reduced to 10 dB, coding provides no improvement; in fact, there is a degradation. One might ask, Why does part (b) manifest a degradation? After all, the same procedure is used for applying the code in both parts of the problem. The answer can be seen in the coded-versus-uncoded pictorial shown in Figure 6.9. Even though Problem 6.5 deals with message-error probability, and Figure 6.9 displays bit-error probability, the following explanation still applies. In all such plots, there is a crossover between the curves (usually at some low value of E_b/N_0). The reason for such crossover (threshold) is that every code system has some fixed error-correcting capability. If there are more errors within a block than the code is capable of correcting, the system will perform poorly. Imagine that E_b/N_0 is continually reduced. What happens at the output of the demodulator? It makes more and more errors. Therefore, such a continual decrease in E_b/N_0 must eventually cause some threshold to be reached where the decoder becomes overwhelmed with errors. When that threshold is crossed, we can interpret the degraded performance as being caused by the redundant bits consuming energy but giving back nothing beneficial in return. Does it strike the reader as a paradox that operating in a region (low values of E_b/N_0), where one would best like to see an error-performance improvement, is where the code makes things worse? There is, however, a class of powerful codes called *turbo codes* that provide error-performance improvements at low values of E_b/N_0; the crossover point is lower for turbo codes compared with conventional codes. (These are treated in Section 8.4.)

Example 6.2 Coded versus Uncoded Performance

Compare the message error probability for a communications link with and without the use of error-correction coding. Assume that the uncoded transmission characteristics are: BPSK modulation, Gaussian noise, $P_r/N_0 = 43{,}776$, data rate $R = 4800$ bits/s. For the coded case, also assume the use of a (15, 11) error-correcting code that is capable of correcting any single-error pattern within a block of 15 bits. Consider that the demodulator makes hard decisions and thus feeds the demodulated code bits directly to the decoder, which in turn outputs an estimate of the original message.

Solution

Following Equation (4.79), let $p_u = Q\sqrt{2E_b/N_0}$ and $p_c = Q\sqrt{2E_c/N_0}$ be the uncoded and coded channel symbol error probabilities, respectively, where E_b/N_0 is the bit energy per noise spectral density and E_c/N_0 is the code-bit energy per noise spectral density.

Without coding

$$\frac{E_b}{N_0} = \frac{P_r}{N_0}\left(\frac{1}{R}\right) = 9.12\ (9.6\ \text{dB})$$

and

$$p_u = Q\left(\sqrt{\frac{2E_b}{N_0}}\right) = Q(\sqrt{18.24}) = 1.02 \times 10^{-5} \tag{6.20}$$

where the following approximation of $Q(x)$ from Equation (3.44) was used:

$$Q(x) \approx \frac{1}{x\sqrt{2\pi}} \exp\left(\frac{-x^2}{2}\right) \qquad \text{for } x > 3$$

The probability that the uncoded message block P_M^u will be received in error is 1 minus the product of the probabilities that each bit will be detected correctly. Thus,

$$\begin{aligned} P_M^u &= 1 - (1 - p_u)^k \\ &= 1 - \underbrace{(1 - p_u)^{11}}_{\substack{\text{probability that all} \\ \text{11 bits in uncoded} \\ \text{block are correct}}} \qquad \underbrace{= 1.12 \times 10^{-4}}_{\substack{\text{probability that at} \\ \text{least 1 bit out of} \\ \text{11 is in error}}} \end{aligned} \tag{6.21}$$

With coding

Assuming a *real-time* communication system such that delay is unacceptable, the channel-symbol rate or code-bit rate R_c is 15/11 times the data bit rate:

$$R_c = 4800 \times \tfrac{15}{11} \approx 6545 \text{ bps}$$

and

$$\frac{E_c}{N_0} = \frac{P_r}{N_0}\left(\frac{1}{R_c}\right) = 6.69\ (8.3\ \text{dB})$$

The E_c/N_0 for each code bit is less than that for the data bit in the uncoded case because the channel-bit rate has increased, but the transmitter power is assumed to be fixed:

$$p_c = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) = Q(\sqrt{13.38} = 1.36 \times 10^{-4} \tag{6.22}$$

It can be seen by comparing the results of Equation 6.20 with those of Equation 6.22 that because redundancy was added, the channel bit-error probability has degraded. More bits must be detected during the same time interval and with the same available power; the performance improvement due to the coding *is not yet apparent.* We now compute the coded message error rate P_M^c, using Equation 6.18:

$$P_M^c = \sum_{j=2}^{n=15} \binom{15}{j} (p_c)^j (1 - p_c)^{15-j}$$

The summation is started with $j = 2$, since the code corrects all single errors within a block of $n = 15$ bits. An approximation is obtained by using only the first term of the summation. For p_c, we use the value calculated in Equation 6.22:

$$P_M^c \approx \binom{15}{2} (p_c)^2 (1 - p_c)^{13} = 1.94 \times 10^{-6} \tag{6.23}$$

By comparing the results of Equation (6.21) with (6.23), we can see that the probability of message error has improved by a factor of 58 due to the error-correcting code used in this example. This example illustrates the typical behavior of all such real-time communication systems using error-correction coding. Added redundancy means faster signaling, less energy per channel symbol, and more errors out of the demodulator. The benefits arise because the behavior of the decoder will (at reasonable values of E_b/N_0) more than compensate for the poor performance of the demodulator.

Linear Block codes

Linear block codes (such as the one described in Example 6.2) are a class of parity-check codes that can be characterized by the (n, k) notation described earlier. The encoder transforms a block of k message digits (a message vector) into a longer block of n codeword digits (a code vector) constructed from a given alphabet of elements. When the alphabet consists of two elements (0 and 1), the code is a binary code comprising binary digits (bits). Our discussion of linear block codes is restricted to binary codes unless otherwise noted.

The k-bit messages form 2^k distinct message sequences, referred to as *k-tuples* (sequences of k digits). The n-bit blocks can form as many as 2^n distinct sequences, referred to as *n-tuples*. The encoding procedure assigns to each of the 2^k message k-tuples *one* of the 2^n n-tuples. A block code represents a one-to-one assignment, whereby the 2^k message k-tuples are *uniquely* mapped into a new set of 2^k codeword n-tuples; the mapping can be accomplished via a look-up table. For *linear codes*, the mapping transformation is, of course *linear*.

6.4.1 Vector Spaces

The set of all binary n-tuples, V_n, is called a *vector space* over the binary field of two elements (0 and 1). The binary field has two operations, addition and multiplication, such that the results of all operations are in the same set of two elements. The arithmetic operations of addition and multiplication are defined by the conventions of the algebraic field [4]. For example, in a binary field, the rules of addition and multiplication are as follows:

Addition	Multiplication
$0 \oplus 0 = 0$	$0 \cdot 0 = 0$
$0 \oplus 1 = 1$	$0 \cdot 1 = 0$
$1 \oplus 0 = 1$	$1 \cdot 0 = 0$
$1 \oplus 1 = 0$	$1 \cdot 1 = 1$

The addition operation, designated with the symbol $\oplus$, is the same modulo-2 operation described in Section 2.9.3. The summation of binary n-tuples always entails modulo-2 addition. However, for notational simplicity the ordinary + sign will often be used.

6.4.2 Vector Subspaces

A subset S of the vector space V_n is called a *subspace* if the following two conditions are met:

1. The all-zeros vector is in S.
2. The sum of any two vectors in S is also in S (known as the *closure property*).

These properties are fundamental for the algebraic characterization of *linear block codes*. Suppose that $\mathbf{V}_i$ and $\mathbf{V}_j$ are two codewords (or code vectors) in an (n, k) binary block code. The code is said to be *linear* if, and only if $(\mathbf{V}_i \oplus \mathbf{V}_j)$ is also a code vector. A linear block code, then, is one in which vectors outside the subspace cannot be created by the addition of legitimate codewords (members of the subspace).

For example, the vector space V_4 is totally populated by the following 2^4 = sixteen 4-tuples:

0000	0001	0010	0011	0100	0101	0110	0111
1000	1001	1010	1011	1100	1101	1110	1111

An example of a subset of V_4 that forms a subspace is

0000 0101 1010 1111

It is easy to verify that the addition of any two vectors in the subspace can only yield one of the other members of the subspace. A set of 2^k n-tuples is called a *linear block code* if, and only if, it is a subspace of the vector space V_n of all n-tuples. Figure 6.10 illustrates, with a simple geometric analogy, the structure behind linear block codes. We can imagine the vector space V_n comprising 2^n n-tuples. Within this vector space there exists a subset of 2^k n-tuples making up a subspace. These 2^k vectors or points, shown "sprinkled" among the more numerous 2^n points, represent the legitimate or allowable codeword assignments. A message is encoded into one of the 2^k allowable code vectors and then transmitted. Because of noise in the channel, a perturbed version of the codeword (one of the other 2^n vectors in the n-tuple space) may be received. If the perturbed vector is not too unlike (not too distant from) the valid codeword, the decoder can decode the message correctly. The basic goals in choosing a particular code, similar to the goals in selecting a set of modulation waveforms, can be stated in the context of Figure 6.10 as follows:

1. We strive for coding efficiency by packing the V_n space with as many codewords as possible. This is tantamount to saying that we only want to expend a *small amount of redundancy* (excess bandwidth).
2. We want the codewords to be as *far apart from one another* as possible, so that even if the vectors experience some corruption during transmission, they may still be correctly decoded, with a high probability.

Well-Know Block Code

6.8.1 Hamming Codes

Hamming codes are a simple class of block codes characterized by the structure

$$(n, k) = (2^m - 1, 2^m - 1 - m) \tag{6.71}$$

where $m = 2, 3, \ldots$. These codes have a minimum distance of 3 and thus, from Equations (6.44) and (6.47), they are capable of correcting all single errors or detecting all combinations of two or fewer errors within a block. Syndrome decoding is especially suited for Hamming codes. In fact, the syndrome can be formed to act as a binary pointer to identify the error location [5]. Although Hamming codes are not very powerful, they belong to a very limited class of block codes known as *perfect* codes, described in Section 6.5.4.

Assuming hard decision decoding, the bit error probability can be written, from Equation (6.46), as

$$P_B \approx \frac{1}{n} \sum_{j=2}^{n} j \binom{n}{j} p^j (1-p)^{n-j} \tag{6.72}$$

where p is the channel symbol error probability (transition probability on the binary symmetric channel). In place of Equation (6.72) we can use the following equivalent equation. Its identity with Equation (6.72) is proven in Appendix D, Equation (D.16):

$$P_B \approx p - p(1-p)^{n-1} \tag{6.73}$$

Figure 6.21 is a plot of the decoded P_B versus channel-symbol error probability, illustrating the comparative performance for different types of block codes. For the Hamming codes, the plots are shown for $m = 3$, 4, and 5, or $(n, k) = (7, 4)$, (15, 11), and (31, 26). For performance over a Gaussian channel using coherently demodulated BPSK, we can express the channel symbol error probability in terms of E_c/N_0, similar to Equation (4.79), as

$$p = Q\left(\sqrt{\frac{2E_c}{N_0}}\right) \tag{6.74}$$

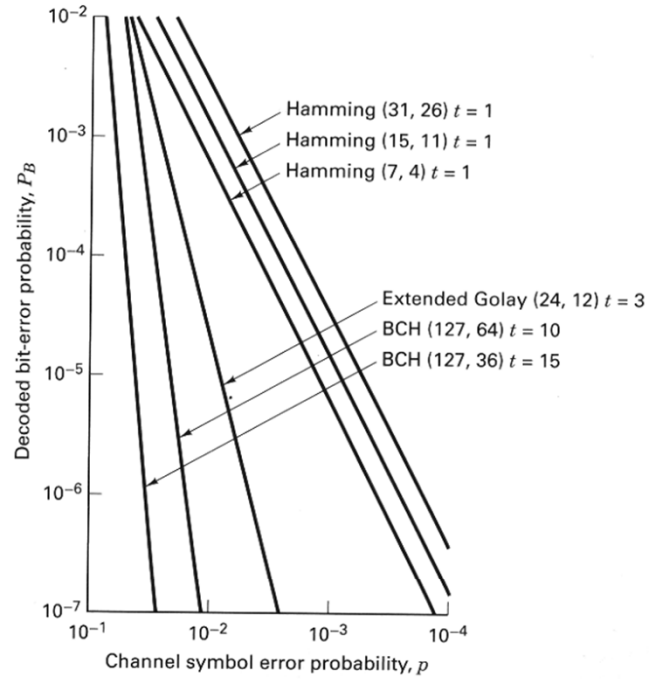


Figure 6.21 Bit error probability versus channel symbol error probability for several block codes.

where E_c/N_0 is the code symbol energy per noise spectral density, and where $Q(x)$ is as defined in Equation (3.43). To relate E_c/N_0 to information bit energy per noise spectral density (E_b/N_0), we use

$$\frac{E_c}{N_0} = \left(\frac{k}{n}\right)\frac{E_b}{N_0} \tag{6.75}$$

For Hamming codes, Equation (6.75) becomes

$$\frac{E_c}{N_0} = \frac{2^m - 1 - m}{2^m - 1}\,\frac{E_b}{N_0} \tag{6.76}$$

Combining Equation (6.73), (6.74), and (6.76), P_B can be expressed as a function of E_b/N_0 for coherently demodulated BPSK over a Gaussian channel. The results are plotted in Figure 6.22 for different types of block codes. For the Hamming codes, plots are shown for $(n, k) = (7, 4)$, (15, 11), and (31, 26).

Well-Know Block Code

Example 6.11 Error Probability for Modulated and Coded Signals

A coded orthogonal BFSK modulated signal is transmitted over a Gaussian channel. The signal is noncoherently detected and hard-decision decoded. Find the decoded bit error probability if the coding is a Hamming (7, 4) block code and the received E_b/N_0 is equal to 20.

Solution

First we need to find E_c/N_0 using Equation (6.75):

$$\frac{E_c}{N_0} = \frac{4}{7}(20) = 11.43$$

Then, for coded noncoherent BFSK, we can relate the probability of a channel symbol error to E_c/N_0, similar to Equation (4.96), as follows

$$p = \frac{1}{2}\exp\left(-\frac{E_c}{2N_0}\right)$$
$$= \frac{1}{2}\exp\left(-\frac{11.43}{2}\right) = 1.6 \times 10^{-3}$$

Using this result in Equation (6.73), we solve for the probability of a decoded bit error, as follows:

$$P_B \approx p - p(1-p)^6 \approx 1.6 \times 10^{-5}$$

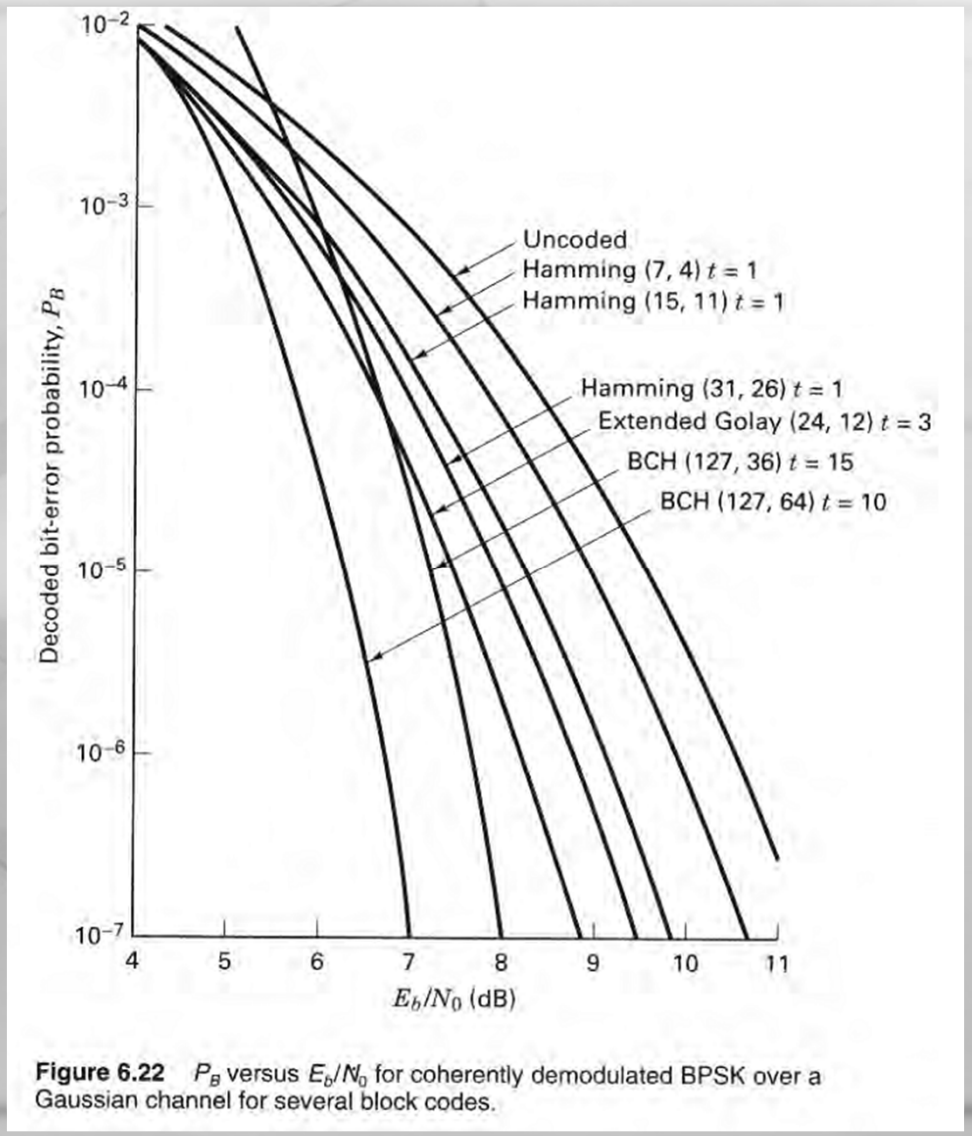


Figure 6.22 P_B versus E_b/N_0 for coherently demodulated BPSK over a Gaussian channel for several block codes.

Well-Know Block Code

6.8.2 Extended Golay Code

One of the more useful block codes is the binary (24, 12) *extended Golay code,* which is formed by adding an overall parity bit to the perfect (23, 12) code, known as the *Golay code.* This added parity bit increases the minimum distance $d_{\min}$ from 7 to 8 and produces a rate $\frac{1}{2}$ code, which is easier to implement (with regard to system clocks) than the rate 12/23 original Golay code. Extended Golay codes are considerably more powerful than the Hamming codes described in the preceding section. The price paid for the improved performance is a more complex decoder, a lower code rate, and hence a larger bandwidth expansion.

Since $d_{\min} = 8$ for the extended Golay code, we see from Equation (6.44) that the code is guaranteed to correct all triple errors. The decoder can additionally be designed to correct *some but not all* four-error patterns. Since only 16.7% of the four-error patterns can be corrected, the decoder, for the sake of simplicity, is usually designed to only correct three-error patterns [5]. Assuming hard decision decoding, the bit error probability for the extended Golay code can be written as a function of the channel symbol error probability p from Equation (6.46), as follows:

$$P_B \approx \frac{1}{24} \sum_{j=4}^{24} j \binom{24}{j} p^j (1-p)^{24-j} \tag{6.77}$$

The plot of Equation (6.77) is shown in Figure 6.21; the error performance of the extended Golay code is seen to be significantly better than that of the Hamming codes. Combining Equations (6.77), (6.74), and (6.75), we can relate P_B versus E_b/N_0 for coherently demodulated BPSK with extended Golay coding over a Gaussian channel. The result is plotted in Figure 6.22.

6.8.3 BCH Codes

Bose–Chadhuri–Hocquenghem (BCH) codes are a generalization of Hamming codes that allow multiple error correction. They are a *powerful class of cyclic codes* that provide a large selection of block lengths, code rates, alphabet sizes, and error-correcting capability. Table 6.4 lists some code generators $\mathbf{g}(x)$ commonly used for the construction of BCH codes [8] for various values of n, k, and t, up to a block length of 255. The coefficients of $\mathbf{g}(x)$ are presented as octal numbers arranged so that when they are converted to binary digits the rightmost digit corresponds to the zero-degree coefficient of $\mathbf{g}(x)$. From Table 6.4, one can easily verify a cyclic code property—the generator polynomial is of degree $n - k$. BCH codes are important because at block lengths of a few hundred, the BCH codes outperform all other block codes with the same block length and code rate. The most commonly used BCH codes employ a binary alphabet and a codeword block length of $n = 2^m - 1$, where $m = 3, 4, \ldots$.

The title of Table 6.4 indicates that the generators shown are for those BCH codes known as *primitive codes.* The term "primitive" is a number-theoretic concept requiring an algebraic development [7, 10–11], which is presented in Section 8.1.4. In Figures 6.21 and 6.22 are plotted error performance curves of two BCH codes (127, 64) and (127, 36), to illustrate comparative performance. Assuming hard decision decoding, the P_B versus channel error probability is shown in Figure 6.21. The P_B versus E_b/N_0 for coherently demodulated BPSK over a Gaussian channel is shown in Figure 6.22. The curves in Figure 6.22 seem to depart from our expectations. They each have the same block size, yet the more redundant (127, 36) code does not exhibit as much coding gain as does the less redundant (127, 64) code. It has been shown that a relatively broad maximum of coding gain versus code rate for fixed n occurs roughly between coding rates of $\frac{1}{3}$ and $\frac{3}{4}$ for BCH codes [12]. Performance over a Gaussian channel degrades substantially at very high or very low rates [11].

Well-Know Block Code

Figure 6.23 represents computed performance of BCH codes [13] using coherently demodulated BPSK with both *hard-* and *soft-decision decoding*. Soft-decision decoding is not usually used with block codes because of its complexity. However, whenever it is implemented, it offers an approximate 2-dB coding gain over hard-decision decoding. For a given code rate, the decoded error probability is known to improve with increasing block length n [4]. Thus, for a given code rate, it is

interesting to consider the block length that would be required for the hard-decision-decoding performance to be comparable to the soft-decision-decoding performance. In Figure 6.23, the BCH codes shown all have code rates of approximately $\frac{1}{2}$. From the figure [13] it appears that for a fixed code rate, the hard-decision-decoded BCH code of length 8 times n or longer has a better performance (for P_B of about 10^{-6} or less) than that of a soft-decision-decoded BCH code of length n. One special subclass of the BCH codes (the discovery of which preceded the BCH codes) is the particularly useful *nonbinary* set called *Reed-Solomon* codes. They are described in Section 8.1.

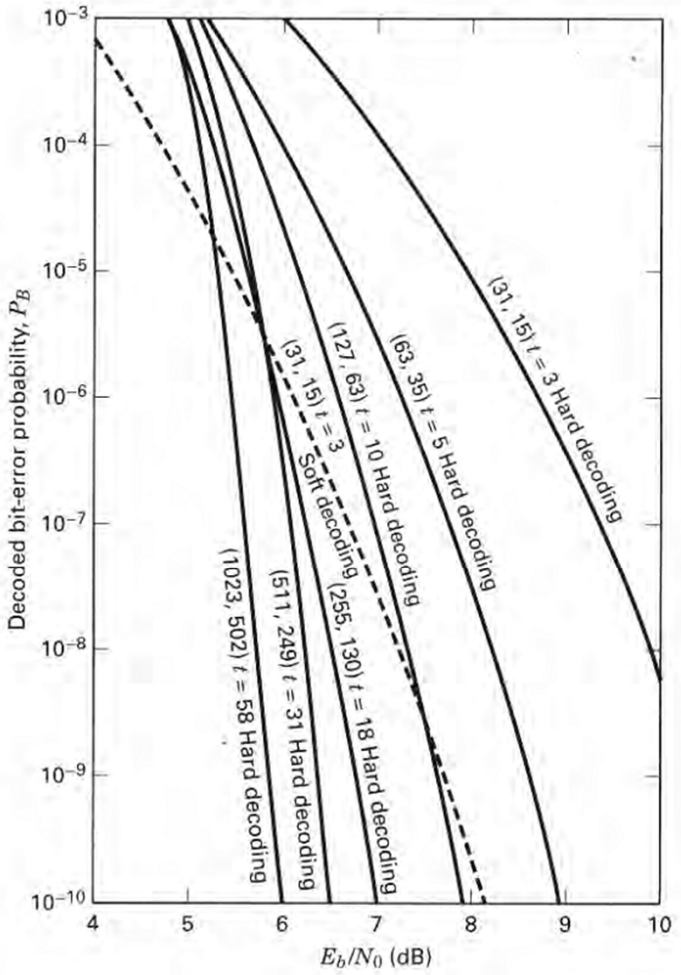


Figure 6.23 P_B versus E_b/N_0 for coherently demodulated BPSK over a Gaussian channel using BCH codes. (Reprinted with permission from L. J. Weng, "Soft and Hard Decoding Performance Comparisons for BCH Codes," *Proc. Int. Conf. Commun.*, 1979, Fig. 3, p. 25.5.5. © 1979 IEEE.)

Conclusion

CONCLUSION

In this chapter we have explored the general goals of channel coding, leading to improved performance (error probability, E_b/N_0, or capacity) at a cost in bandwidth. We partitioned channel coding into two study groups: waveform coding and structured sequences. Waveform coding represents a transformation of waveforms into improved waveforms, such that the distance properties are improved over those of the original waveforms. Structured sequences involve the addition of redundant digits to the data, such that the redundant digits can then be employed for detecting and/or correcting specific error patterns.

We also closely examined linear block codes. Geometric analogies can be drawn between the coding and modulation disciplines. They both seek to pack the signal space efficiently and to maximize the distance between signals in the signaling set. Within block codes, we looked at cyclic codes, which are relatively easy to implement using modern integrated circuit techniques. We considered the polynomial representation of codes and the correspondence between the polynomial structure, the necessary algebraic operations, and the hardware implementation. Finally, we looked at performance details of some of the well-known block codes. Other coding subjects are treated in later chapters. In Chapter 7 we study the large class of convolutional codes; in Chapter 8 we discuss Reed–Solomon codes, concatenated codes, and turbo codes; and in Chapter 9 we examine trellis-coded modulation.

Problems and Questions

- Problems

6.2. Calculate the probability of message error for a 12-bit data sequence encoded with a (24, 12) linear block code. Assume that the code corrects all 1-bit and 2-bit error patterns and assume that it corrects no error patterns with more than two errors. Also, assume that the probability of a channel symbol error is 10^{-3}.

6.3. Consider a (127, 92) linear block code capable of triple error corrections.

(a) What is the probability of message error for an uncoded block of 92 bits if the channel symbol error probability is 10^{-3}?

(b) What is the probability of message error when using the (127, 92) block code if the channel symbol error probability of 10^{-3}?

6.4. Calculate the improvement in probability of message error relative to an uncoded transmission for a (24, 12) double-error-correcting linear block code. Assume that coherent BPSK modulation is used and that the received $E_b/N_0 = 10$ dB.

6.5. Consider a (24, 12) linear block code capable of double-error corrections. Assume that a noncoherently detected binary orthogonal frequency-shift keying (BFSK) modulation format is used and that the received $E_b/N_0 = 14$ dB.

(a) Does the code provide any improvement in probability of message error? If it does, how much? If it does not, explain why not.

(b) Repeat part (a) with $E_b/N_0 = 10$ dB.

Problems and Questions

- Questions

QUESTIONS

6.1. Describe four types of trade-offs that can be accomplished by using an error-correcting code. (See Section 6.3.4.)

6.2. In a *real-time* communication system, achieving coding gain by adding redundancy costs *bandwidth*. What is the usual cost for achieving coding gain in a *non-real-time* communication system? (See Section 6.3.4.2.)

6.3. In a real-time communication system, added redundancy means faster signaling, less energy per channel symbol, and more errors out of the demodulator. In the face of such degradation, explain how coding gain is achieved. (See Example 6.2.)

6.4. Why do error-correcting codes typically yield error-performance degradation at low values of E_b/N_0? (See Section 6.3.4.6.)

6.5. Describe the process of syndrome testing, error detection and correction in the context of a medical analogy. (See Section 6.4.8.4.)

6.6. Of what use is the *standard array* in understanding a block code, and in evaluating its capability? (See Section 6.6.5.)