1

Coding for Reliable Digital Transmission and Storage

1.1 INTRODUCTION

In recent years, there has been an increasing demand for efficient and reliable digital data transmission and storage systems. This demand has been accelerated by the emergence of large-scale, high-speed data networks for the exchange, processing, and storage of digital information in the military, governmental, and private spheres. A merging of communications and computer technology is required in the design of these systems. A major concern of the designer is the control of errors so that reliable reproduction of data can be obtained.

In 1948, Shannon [1] demonstrated in a landmark paper that, by proper encoding of the information, errors induced by a noisy channel or storage medium can be reduced to any desired level without sacrificing the rate of information transmission or storage. Since Shannon's work, a great deal of effort has been expended on the problem of devising efficient encoding and decoding methods for error control in a noisy environment. Recent developments have contributed toward achieving the reliability required by today's high-speed digital systems, and the use of coding for error control has become an integral part in the design of modern communication systems and digital computers.

The transmission and storage of digital information have much in common. They both transfer data from an information source to a destination (or user). A typical transmission (or storage) system may be represented by the block diagram shown in Figure 1.1. The *information source* can be either a person or a machine (e.g., a digital computer). The source output, which is to be communicated to the destination, can be either a continuous waveform or a sequence of discrete symbols.

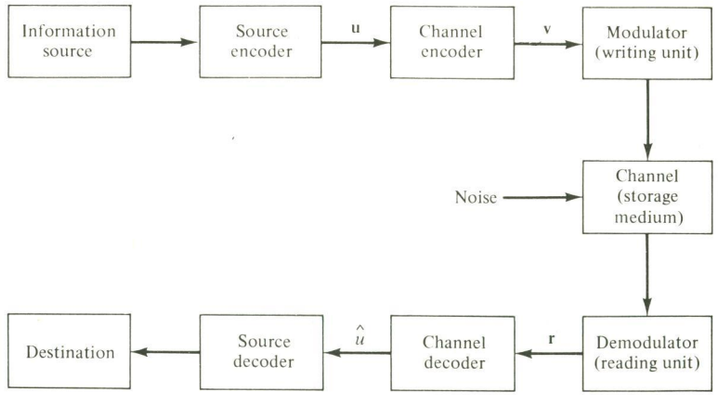

Figure 1.1 Block diagram of a typical data transmission or storage system.

The *source encoder* transforms the source output into a sequence of binary digits (bits) called the *information sequence* **u**. In the case of a continuous source, this involves analog-to-digital (A/D) conversion. The source encoder is ideally designed so that (1) the number of bits per unit time required to represent the source output is minimized, and (2) the source output can be reconstructed from the information sequence **u** without ambiguity. The subject of source coding is not discussed in this book. For a thorough treatment of this important topic, see References 2 and 3.

The *channel encoder* transforms the information sequence **u** into a discrete *encoded sequence* **v** called a *code word*. In most instances **v** is also a binary sequence, although in some applications nonbinary codes have been used. The design and implementation of channel encoders to combat the noisy environment in which code words must be transmitted or stored is one of the major topics of this book.

Discrete symbols are not suitable for transmission over a physical channel or recording on a digital storage medium. The *modulator* (or *writing unit*) transforms each output symbol of the channel encoder into a waveform of duration T seconds which is suitable for transmission (or recording). This waveform enters the *channel* (or *storage medium*) and is corrupted by noise. Typical transmission channels include telephone lines, high-frequency radio links, telemetry links, microwave links, satellite links, and so on. Typical storage media include core and semiconductor memories, magnetic tapes, drums, disk files, optical memory units, and so on. Each of these examples is subject to various types of noise disturbances. On a telephone line, the disturbance may come from switching impulse noise, thermal noise, crosstalk from other lines, or lightning. On magnetic tape, surface defects are regarded as a noise disturbance. The *demodulator* (or *reading unit*) processes each received waveform of duration T and produces an output that may be discrete (quantized) or continuous (unquantized). The sequence of demodulator outputs corresponding to the encoded sequence **v** is called the *received sequence* **r**.

The *channel decoder* transforms the received sequence **r** into a binary sequence $\hat{\mathbf{u}}$ called the *estimated sequence*. The decoding strategy is based on the rules of channel encoding and the noise characteristics of the channel (or storage medium). Ideally, $\hat{\mathbf{u}}$

will be a replica of the information sequence **u**, although the noise may cause some *decoding errors*. Another major topic of this book is the design and implementation of channel decoders that minimize the probability of decoding error.

The *source decoder* transforms the *estimated sequence* $\hat{\mathbf{u}}$ into an *estimate* of the source output and delivers this estimate to the *destination*. When the source is continuous, this involves digital-to-analog (D/A) conversion. In a well-designed system, the estimate will be a faithful reproduction of the source output except when the channel (or storage medium) is very noisy.

To focus attention on the channel encoder and channel decoder, (1) the information source and source encoder are combined into a *digital source* with output **u**; (2) the modulator (or writing unit), the channel (or storage medium), and the demodulator (or reading unit) are combined into a *coding channel* with input **v** and output **r**; and (3) the source decoder and destination are combined into a *digital sink* with input $\hat{\mathbf{u}}$. A simplified block diagram is shown in Figure 1.2.

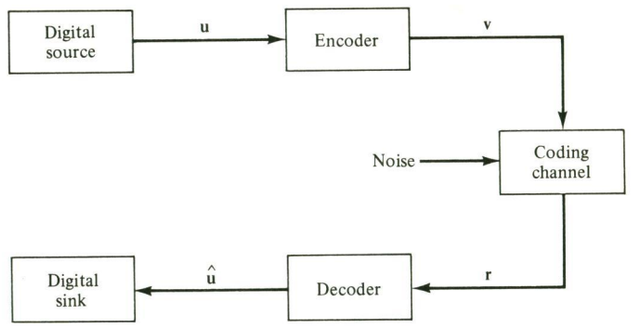

Figure 1.2 Simplified model of a coded system.

The major engineering problem that is addressed in this book is to design and implement the channel encoder/decoder pair such that (1) information can be transmitted (or recorded) in a noisy environment as fast as possible, (2) reliable reproduction of the information can be obtained at the output of the channel decoder, and (3) the cost of implementing the encoder and decoder falls within acceptable limits.

1.2 TYPES OF CODES

There are two different types of codes in common use today, block codes and convolutional codes. The encoder for a block code divides the information sequence into message blocks of k information bits each. A message block is represented by the binary k-tuple $\mathbf{u} = (u_1, u_2, \ldots, u_k)$ called a *message*. (In block coding, the symbol **u** is used to denote a k-bit message rather than the entire information sequence.) There are a total of 2^k different possible messages. The encoder transforms each message **u** independently into an n-tuple $\mathbf{v} = (v_1, v_2, \ldots, v_n)$ of discrete symbols called a *code word*. (In block coding, the symbol **v** is used to denote an n-symbol block rather than the entire encoded sequence.) Therefore, corresponding to the 2^k different possible messages, there are 2^k different possible code words at the encoder

output. This set of 2^k code words of length n is called an (n, k) *block code*. The ratio $R = k/n$ is called the *code rate*, and can be interpreted as the number of information bits entering the encoder per transmitted symbol. Since the n-symbol output code word depends only on the corresponding k-bit input message, the encoder is memoryless, and can be implemented with a combinational logic circuit.

In a binary code, each code word $\mathbf{v}$ is also binary. Hence, for a binary code to be useful (i.e., to have a different code word assigned to each message), $k \leq n$ or $R \leq 1$. When $k < n$, $n - k$ redundant bits can be added to each message to form a code word. These redundant bits provide the code with the capability of combating the channel noise. For a fixed code rate R, more redundant bits can be added by increasing the block length n of the code while holding the ratio k/n constant. How to choose these redundant bits to achieve reliable transmission over a noisy channel is the major problem in designing the encoder. An example of a binary block code with $k = 4$ and $n = 7$ is shown in Table 1.1. Chapters 3 through 9 are devoted to the analysis and design of block codes for controlling errors in a noisy environment.

TABLE 1.1 BINARY BLOCK CODE WITH $k = 4$ AND $n = 7$

Messages	Code words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

The encoder for a convolutional code also accepts k-bit blocks of the information sequence $\mathbf{u}$ and produces an encoded sequence (code word) $\mathbf{v}$ of n-symbol blocks. (In convolutional coding, the symbols $\mathbf{u}$ and $\mathbf{v}$ are used to denote sequences of blocks rather than a single block.) However, each encoded block depends not only on the corresponding k-bit message block at the same time unit, but also on m previous message blocks. Hence, the encoder has a *memory order* of m. The set of encoded sequences produced by a k-input, n-output encoder of memory order m is called an (n, k, m) *convolutional code*. The ratio $R = k/n$ is called the *code rate*. Since the encoder contains memory, it must be implemented with a sequential logic circuit.

In a binary convolutional code, redundant bits for combating the channel noise

can be added to the information sequence when $k < n$ or $R < 1$. Typically, k and n are small integers and more redundancy is added by increasing the memory order m of the code while holding k and n, and hence the code rate R, fixed. How to use the memory to achieve reliable transmission over a noisy channel is the major problem in designing the encoder. An example of a binary convolutional encoder with $k = 1$, $n = 2$, and $m = 2$ is shown in Figure 1.3. As an illustration of how code words are generated, consider the information sequence $\mathbf{u} = (1\ 1\ 0\ 1\ 0\ 0\ 0\ \ldots)$, where the leftmost bit is assumed to enter the encoder first. Using the rules of exclusive-or addition, and assuming that the multiplexer takes the first encoded bit from the top output, it is easy to see that the encoded sequence is $\mathbf{v} = (1\ 1,\ 1\ 0,\ 1\ 0,\ 0\ 0,\ 0\ 1,\ 1\ 1,\ 0\ 0,\ 0\ 0,\ 0\ 0,\ \ldots)$. Chapters 10 through 14 are devoted to the analysis and design of convolutional codes for controlling errors in a noisy environment.

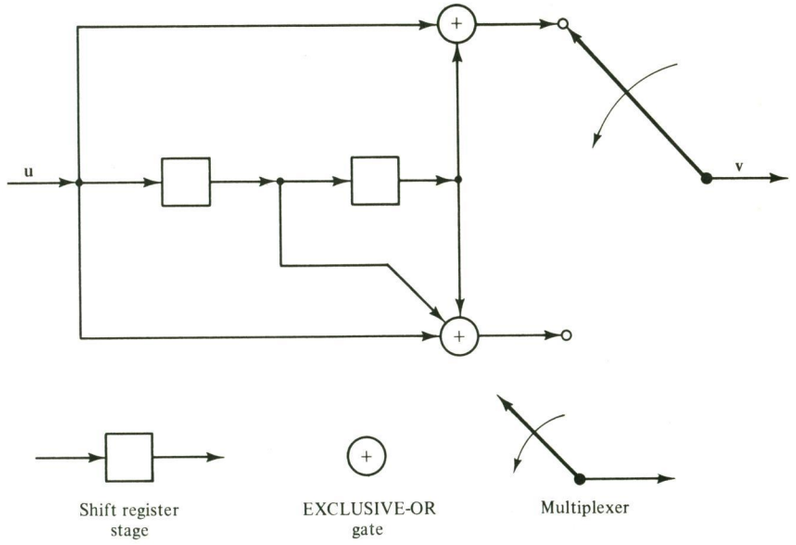

Figure 1.3 Binary convultional encoder with $k = 1$, $n = 2$, and $m = 2$.

1.3 MODULATION AND DEMODULATION

The modulator in a communication system must select a waveform of duration T seconds, which is suitable for transmission, for each encoder output symbol. In the case of a binary code, the modulator must generate one of two signals, $s_0(t)$ for an encoded "0" or $s_1(t)$ for an encoded "1." For a wideband channel, the optimum choice of signals is

$$s_0(t) = \sqrt{\frac{2E}{T}} \sin\left(2\pi f_0 t + \frac{\pi}{2}\right), \qquad 0 \leq t \leq T$$
$$s_1(t) = \sqrt{\frac{2E}{T}} \sin\left(2\pi f_0 t - \frac{\pi}{2}\right), \qquad 0 \leq t \leq T, \tag{1.1}$$

where f_0 is a multiple of $1/T$ and E is the energy of each signal. This is called *binary-phase-shift-keyed* (BPSK) modulation, since the transmitted signal is a sine-wave pulse whose phase is either $+\pi/2$ or $-\pi/2$, depending on the encoder output. The BPSK modulated waveform corresponding to the code word $\mathbf{v} = (1\ 1\ 0\ 1\ 0\ 0\ 0)$ in the code of Table 1.1 is shown in Figure 1.4.

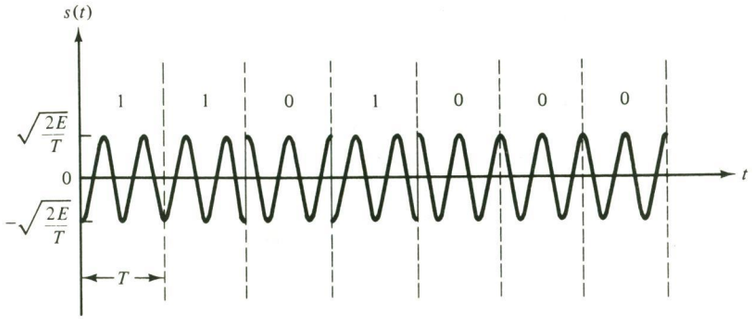

Figure 1.4 BPSK modulated waveform corresponding to the code word $\mathbf{v} = (1\ 1\ 0\ 1\ 0\ 0\ 0)$.

A common form of noise disturbance present in any communication system is *additive white Gaussian noise* (AWGN). If the transmitted signal is $s(t)$ [$= s_0(t)$ or $s_1(t)$], the received signal is

$$r(t) = s(t) + n(t), \tag{1.2}$$

where $n(t)$ is a Gaussian random process with one-sided *power spectral density* (PSD) N_0. Other forms of noise are also present in many systems. For example, in a communication system subject to multipath transmission, the received signal is observed to fade (lose strength) during certain time intervals. This fading can be modeled as a multiplicative noise component on the signal $s(t)$.

The demodulator must produce an output corresponding to the received signal in each T-second interval. This output may be a real number or one of a discrete set of preselected symbols, depending on the demodulator design. An optimum demodulator always includes a matched filter or correlation detector followed by a sampling switch. For BPSK modulation with coherent detection the sampled output is a real number,

$$\rho = \int_0^T r(t)\sqrt{\frac{2E}{T}} \sin\left(2\pi f_0 t + \frac{\pi}{2}\right) dt. \tag{1.3}$$

The sequence of unquantized demodulator outputs can be passed on directly to the channel decoder for processing. In this case, the channel decoder must be capable of handling analog inputs; that is, it must be an *analog decoder*. A much more common approach to decoding is to quantize the continuous detector output ρ into one of a finite number Q of discrete output symbols. In this case, the channel decoder has discrete inputs; that is, it must be a *digital decoder*. Almost all coded communication systems use some form of digital decoding.

If the detector output in a given interval depends only on the transmitted signal

in that interval, and not on any previous transmission, the channel is said to be *memoryless*. In this case, the combination of an M-ary input modulator, the physical channel, and a Q-ary output demodulator can be modeled as a *discrete memoryless channel* (DMC). A DMC is completely described by a set of *transition probabilities* $P(j|i)$, $0 \le i \le M-1$, $0 \le j \le Q-1$, where i represents a modulator input symbol, j represents a demodulator output symbol, and $P(j|i)$ is the probability of receiving j given that i was transmitted. As an example, consider a communication system in which (1) binary modulation is used ($M = 2$), (2) the amplitude distribution of the noise is symmetric, and (3) the demodulator output is quantized to $Q = 2$ levels. In this case a particularly simple and practically important channel model, called the *binary symmetric channel* (BSC), results. The transition probability diagram for a BSC is shown in Figure 1.5(a). Note that the transition probability p completely describes the channel.

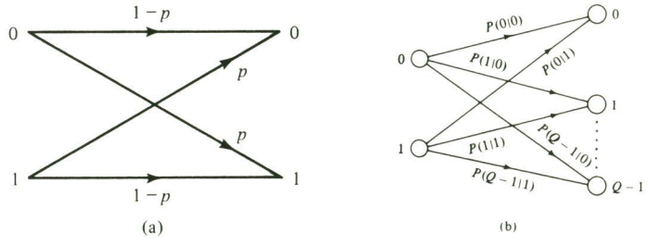

Figure 1.5 Transition probability diagrams: (a) binary symmetric channel; (b) binary-input, Q-ary-output discrete memoryless channel.

The transition probability p can be calculated from a knowledge of the signals used, the probability distribution of the noise, and the output quantization threshold of the demodulator. When BPSK modulation is used on an AWGN channel with optimum coherent detection and binary output quantization, the BSC transition probability is just the BPSK bit error probability for equally likely signals given by

$$p = Q\left(\sqrt{\frac{2E}{N_0}}\right), \tag{1.4}$$

where $Q(x) \triangleq (1/\sqrt{2\pi}) \int_x^\infty e^{-y^2/2}\, dy$ is the *complementary error function* of Gaussian statistics. An upper bound on $Q(x)$ which will be used later in evaluating the error performance of codes on a BSC is

$$Q(x) \le \tfrac{1}{2} e^{-x^2/2}, \qquad x \ge 0. \tag{1.5}$$

When binary coding is used, the modulator has only binary inputs ($M = 2$). Similarly, when binary demodulator output quantization is used ($Q = 2$), the decoder has only binary inputs. In this case, the demodulator is said to make *hard decisions*. Most coded digital communication systems, whether block or convolutional, use binary coding with hard-decision decoding, owing to the resulting simplicity of

implementation compared to nonbinary systems. However, some binary coded systems do not use hard decisions at the demodulator output. When $Q > 2$ (or the output is left unquantized) the demodulator is said to make *soft decisions*. In this case the decoder must accept multilevel (or analog) inputs. Although this makes the decoder more difficult to implement, soft-decision decoding offers significant performance improvement over hard-decision decoding, as discussed in Chapter 11. A transition probability diagram for a soft-decision DMC with $M = 2$ and $Q > 2$ is shown in Figure 1.5(b). This is the appropriate model for a binary-input AWGN channel with finite output quantization. The transition probabilities can be calculated from a knowledge of the signals used, the probability distribution of the noise, and the output quantization thresholds of the demodulator in a manner similar to the calculation of the BSC transition probability p. For a more thorough treatment of the calculation of DMC transition probabilities, see References 4 and 5.

If the detector output in a given interval depends on the transmitted signal in previous intervals as well as the transmitted signal in the present interval, the channel is said to have *memory*. A fading channel is a good example of a channel with memory, since the multipath transmission destroys the independence from interval to interval. Appropriate models for channels with memory are difficult to construct, and coding for these channels is normally done on an ad hoc basis.

Two important and related parameters in any digital communication system are the speed of information transmission and the bandwidth of the channel. Since one encoded symbol is transmitted every T seconds, the *symbol transmission rate* (baud rate) is $1/T$. In a coded system, if the code rate is $R = k/n$, k information bits correspond to the transmission of n symbols, and the *information transmission rate* (data rate) is R/T bits per second (bps). In addition to signal modification due to the effects of noise, all communication channels are subject to signal distortion due to bandwidth limitations. To minimize the effect of this distortion on the detection process, the channel should have a *bandwidth* W of roughly $1/2T$ hertz (Hz).[1] In an uncoded system ($R = 1$), the data rate is $1/T = 2W$, and is limited by the channel bandwidth. In a binary-coded system, with a code rate $R < 1$, the data rate is $R/T = 2RW$, and is reduced by the factor R compared to an uncoded system. Hence, to maintain the same data rate as the uncoded system, the coded system requires a *bandwidth expansion* by a factor of $1/R$. This is characteristic of binary-coded systems: they require some bandwidth expansion to maintain a constant data rate. If no additional bandwidth is available without undergoing severe signal distortion, binary coding is not feasible, and other means of reliable communication must be sought.[2]

1.4 MAXIMUM LIKELIHOOD DECODING

A block diagram of a coded system on an AWGN channel with finite output quantization is shown in Figure 1.6. In a block-coded system, the source output $\mathbf{u}$ represents a k-bit message, the encoder output $\mathbf{v}$ represents an n-symbol code word, the demodulator output $\mathbf{r}$ represents the corresponding Q-ary received n-tuple, and the decoder output $\hat{\mathbf{u}}$ represents the k-bit estimate of the encoded message. In a convolutional coded system, $\mathbf{u}$ represents a sequence of kL information bits and $\mathbf{v}$ represents a code word containing $N \triangleq nL + nm = n(L + m)$ symbols, where kL is the length of the information sequence and N is the length of the code word. The additional nm encoded symbols are produced after the last block of information bits has entered the encoder. This is due to the m time unit memory of the encoder, and is discussed more fully in Chapter 10. The demodulator output $\mathbf{r}$ is a Q-ary received N-tuple, and the decoder output $\hat{\mathbf{u}}$ is a kL-bit estimate of the information sequence.

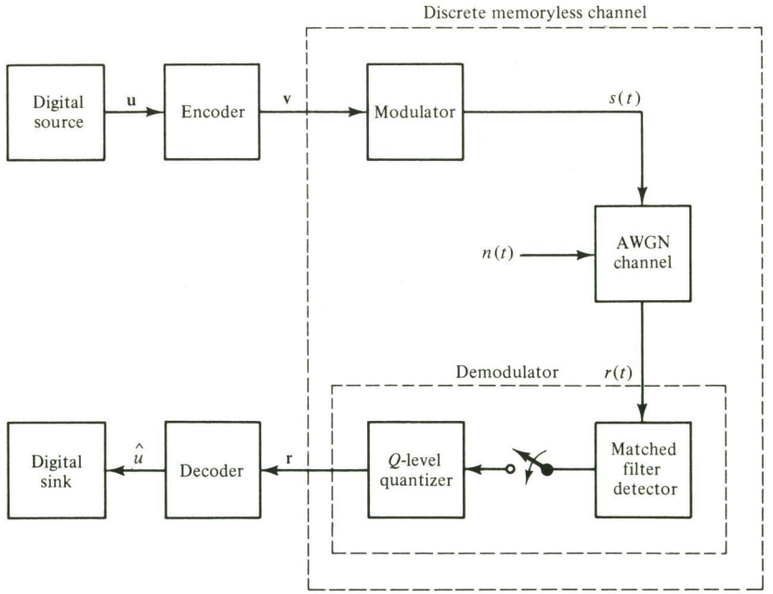

Figure 1.6 Coded system on an additive white Gaussian noise channel.

The decoder must produce an estimate $\hat{\mathbf{u}}$ of the information sequence $\mathbf{u}$ based on the received sequence $\mathbf{r}$. Equivalently, since there is a one-to-one correspondence between the information sequence $\mathbf{u}$ and the code word $\mathbf{v}$, the decoder can produce an estimate $\hat{\mathbf{v}}$ of the code word $\mathbf{v}$. Clearly, $\hat{\mathbf{u}} = \mathbf{u}$ if and only if $\hat{\mathbf{v}} = \mathbf{v}$. A *decoding rule* is a strategy for choosing an estimated code word $\hat{\mathbf{v}}$ for each possible received sequence $\mathbf{r}$. If the code word $\mathbf{v}$ was transmitted, a *decoding error* occurs if and only if $\hat{\mathbf{v}} \neq \mathbf{v}$. Given that $\mathbf{r}$ is received, the *conditional error probability of the decoder* is defined as

$$P(E|\mathbf{r}) \triangleq P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r}). \tag{1.6}$$

The *error probability of the decoder* is then given by

$$P(E) = \sum_{\mathbf{r}} P(E|\mathbf{r})P(\mathbf{r}). \tag{1.7}$$

$P(\mathbf{r})$ is independent of the decoding rule used since $\mathbf{r}$ is produced prior to decoding. Hence, an optimum decoding rule [i.e., one that minimizes $P(E)$] must minimize $P(E|\mathbf{r}) = P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ for all $\mathbf{r}$. Since minimizing $P(\hat{\mathbf{v}} \neq \mathbf{v}|\mathbf{r})$ is equivalent to maximiz-

[1] The exact bandwidth required depends on the shape of the signal waveform, the acceptable limits of distortion, and the definition of bandwidth.

[2] This does not preclude the use of coding, but requires only that a larger set of signals be found. See References 4 to 6.

ing $P(\hat{\mathbf{v}} = \mathbf{v}|\mathbf{r})$, $P(E|\mathbf{r})$ is minimized for a given $\mathbf{r}$ by choosing $\hat{\mathbf{v}}$ as the code word $\mathbf{v}$ which maximizes

$$P(\mathbf{v}|\mathbf{r}) = \frac{P(\mathbf{r}|\mathbf{v})P(\mathbf{v})}{P(\mathbf{r})}, \tag{1.8}$$

that is, $\hat{\mathbf{v}}$ is chosen as the most likely code word given that $\mathbf{r}$ is received. If all information sequences, and hence all code words, are equally likely [i.e., $P(\mathbf{v})$ is the same for all $\mathbf{v}$], maximizing (1.8) is equivalent to maximizing $P(\mathbf{r}|\mathbf{v})$. For a DMC

$$P(\mathbf{r}|\mathbf{v}) = \prod_i P(r_i|v_i), \tag{1.9}$$

since for a memoryless channel each received symbol depends only on the corresponding transmitted symbol. A decoder that chooses its estimate to maximize (1.9) is called a *maximum likelihood decoder* (MLD). Since $\log x$ is a monotone increasing function of x, maximizing (1.9) is equivalent to maximizing the *log-likelihood function*

$$\log P(\mathbf{r}|\mathbf{v}) = \sum_i \log P(r_i|v_i). \tag{1.10}$$

An MLD for a DMC then chooses $\hat{\mathbf{v}}$ as the code word $\mathbf{v}$ that maximizes the sum in (1.10). If the code words are not equally likely, an MLD is not necessarily optimum, since the conditional probabilities $P(\mathbf{r}|\mathbf{v})$ must be weighted by the code word probabilities $P(\mathbf{v})$ to determine which code word maximizes $P(\mathbf{v}|\mathbf{r})$. However, in many systems, the code word probabilities are not known exactly at the receiver, making optimum decoding impossible, and an MLD then becomes the best feasible decoding rule.

Now consider specializing the MLD decoding rule to the BSC. In this case $\mathbf{r}$ is a binary sequence which may differ from the transmitted code word $\mathbf{v}$ in some positions because of the channel noise. When $r_i \neq v_i$, $P(r_i|v_i) = p$, and when $r_i = v_i$, $P(r_i|v_i) = 1 - p$. Let $d(\mathbf{r}, \mathbf{v})$, be the distance between $\mathbf{r}$ and $\mathbf{v}$ (i.e., the number of positions in which $\mathbf{r}$ and $\mathbf{v}$ differ). For a block code of length n, (1.10) becomes

$$\begin{aligned} \log P(\mathbf{r}|\mathbf{v}) &= d(\mathbf{r}, \mathbf{v}) \log p + [n - d(\mathbf{r}, \mathbf{v})] \log (1 - p) \\ &= d(\mathbf{r}, \mathbf{v}) \log \frac{p}{1-p} + n \log (1 - p). \end{aligned} \tag{1.11}$$

[For a convolutional code, n in (1.11) is replaced by N.] Since $\log [p/(1-p)] < 0$ for $p < \frac{1}{2}$ and $n \log (1 - p)$ is a constant for all $\mathbf{v}$, the MLD decoding rule for the BSC *chooses* $\hat{\mathbf{v}}$ *as the code word* $\mathbf{v}$ *which minimizes the distance* $d(\mathbf{r}, \mathbf{v})$ *between* $\mathbf{r}$ *and* $\mathbf{v}$; *that is, it chooses the code word that differs from the received sequence in the fewest number of positions*. Hence, an MLD for the BSC is sometimes called a *minimum distance decoder*.

The capability of a noisy channel to transmit information reliably was determined by Shannon [1] in his original work. This result, called the *noisy channel coding theorem*, states that every channel has a *channel capacity* C, and that for any rate $R < C$, there exists codes of rate R which, with maximum likelihood decoding, have an arbitrarily small decoding error probability $P(E)$. In particular, for any $R < C$, there exists block codes of length n such that

$$P(E) \leq 2^{-nE_b(R)}, \tag{1.12}$$

and there exists convolutional codes of *memory order* m such that

$$P(E) \leq 2^{-(m+1)nE_c(R)} = 2^{-n_A E_c(R)}, \tag{1.13}$$

where $n_A \triangleq (m + 1)n$ is called the code *constraint length*. $E_b(R)$ and $E_c(R)$ are positive functions of R for $R < C$ and are completely determined by the channel characteristics. The bound of (1.12) implies that arbitrarily small error probabilities are achievable with block coding for any fixed $R < C$ by increasing the block length n while holding the ratio k/n constant. The bound of (1.13) implies that arbitrarily small error probabilities are achievable with convolutional coding for any fixed $R < C$ by increasing the constraint length n_A (i.e., by increasing the memory order m while holding k and n constant).

The noisy channel coding theorem is based on an argument called *random coding*. The bound obtained is actually on the average error probability of the ensemble of all codes. Since some codes must perform better than the average, the noisy channel coding theorem guarantees the existence of codes satisfying (1.12) and (1.13), but does not indicate how to construct these codes. Furthermore, to achieve very low error probabilities for block codes of fixed rate $R < C$, long block lengths are needed. This requires that the number of code words $2^k = 2^{nR}$ must be very large. Since a MLD must compute $\log P(\mathbf{r}|\mathbf{v})$ for each code word, and then choose the code word that gives the maximum, the number of computations that must be performed by a MLD becomes excessively large. For convolutional codes, low error probabilities require a large memory order m. As will be seen in Chapter 11, a MLD for convolutional codes requires approximately 2^{km} computations to decode each block of k information bits. This, too, becomes excessively large as m increases. Hence, it is impractical to achieve very low error probabilities with maximum likelihood decoding. Therefore, two major problems are encountered when designing a coded system to achieve low error probabilities: (1) to construct good long codes whose performance with maximum likelihood decoding would satisfy (1.12) and (1.13), and (2) to find easily implementable methods of encoding and decoding these codes such that their actual performance is close to what could be achieved with maximum likelihood decoding. The remainder of this book is devoted to finding solutions to these two problems.

1.5 TYPES OF ERRORS

On memoryless channels, the noise affects each transmitted symbol independently. As an example, consider the BSC whose transition diagram is shown in Figure 1.5(a). Each transmitted bit has a probability p of being received incorrectly and a probability $1 - p$ of being received correctly, independently of other transmitted bits. Hence transmission errors occur randomly in the received sequence, and memoryless channels are called *random-error channels*. Good examples of random-error channels are the deep-space channel and many satellite channels. Most line-of-sight transmission facilities, as well, are affected primarily by random errors. The codes devised for correcting random errors are called *random-error-correcting codes*. Most of the codes presented in this book are random-error-correcting codes. In particular, Chapters 3 through 8 and 10 through 13 are devoted to codes of this type.

On channels with memory, the noise is not independent from transmission to transmission. A simplified model of a channel with memory is shown in Figure 1.7. This model contains two states, a "good state," in which transmission errors occur infrequently, $p_1 \approx 0$, and a "bad state," in which transmission errors are highly

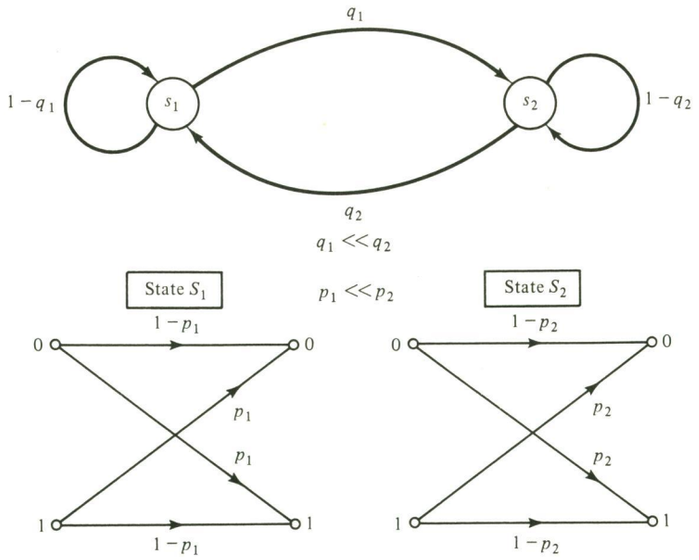

Figure 1.7 Simplified model of a channel with memory.

probable, $p_2 \approx 0.5$. The channel is in the good state most of the time, but on occasion shifts to the bad state due to a change in the transmission characteristic of the channel (e.g., a "deep fade" caused by multipath transmission). As a consequence, transmission errors occur in clusters or bursts because of the high transition probability in the bad state, and channels with memory are called *burst-error channels*. Examples of burst-error channels are radio channels, where the error bursts are caused by signal fading due to multipath transmission, wire and cable transmission, which is affected by impulsive switching noise and crosstalk, and magnetic recording, which is subject to tape dropouts due to surface defects and dust particles. The codes devised for correcting burst errors are called *burst-error-correcting codes*. Sections 9.1 to 9.5 and 14.1 to 14.3 are devoted to codes of this type.

Finally, some channels contain a combination of both random and burst errors. These are called *compound channels*, and codes devised for correcting errors on these channels are called *burst-and-random-error-correcting codes*. Sections 9.6, 9.7, and 14.4 are devoted to codes of this type.

1.6 ERROR CONTROL STRATEGIES

The block diagram shown in Figure 1.1 represents a one-way system. The transmission (or recording) is strictly in one direction, from transmitter to receiver. Error control for a one-way system must be accomplished using *forward error correction* (FEC), that is, by employing error-correcting codes that automatically correct errors detected at the receiver. Examples are magnetic tape storage systems, in which the

information recorded on tape may be replayed weeks or even months after it is recorded, and deep-space communication systems, where the relatively simple encoding equipment can be placed aboard the spacecraft, but the much more complex decoding procedure must be performed on earth. Most of the coded systems in use today employ some form of FEC, even if the channel is not strictly one-way. This book is devoted mostly to the analysis and design of FEC systems. Applications of FEC to storage and communication systems are presented in Chapter 16 and Sections 17.1 to 17.4.

In some cases, a transmission system can be two-way; that is, information can be sent in both directions and the transmitter also acts as a receiver (a transceiver), and vice versa. Examples of two-way systems are telephone channels and some satellite communication systems. Error control for a two-way system can be accomplished using error detection and retransmission, called *automatic repeat request* (ARQ). In an ARQ system, when errors are detected at the receiver, a request is sent for the transmitter to repeat the message, and this continues until the message is received correctly.

There are two types of ARQ systems: stop-and-wait ARQ and continuous ARQ. With *stop-and-wait ARQ*, the transmitter sends a code word to the receiver and waits for a positive (ACK) or negative (NAK) acknowledgment from the receiver. If ACK is received (no errors detected), the transmitter sends the next code word. If NAK is received (errors detected), it resends the preceding code word. When the noise is persistent, the same code word may be retransmitted several times before it is correctly received and acknowledged.

With *continuous ARQ*, the transmitter sends code words to the receiver continuously and receives acknowledgments continuously. When a NAK is received, the transmitter begins a retransmission. It may back up to the code word in error and resend that word plus the words that follow it. This is called *go-back-N ARQ*. Alternatively, the transmitter may simply resend only those code words that are acknowledged negatively. This is known as *selective-repeat ARQ*. Selective-repeat ARQ is more efficient than go-back-N ARQ, but requires more logic and buffering.

Continuous ARQ is more efficient than stop-and-wait ARQ, but it is also more expensive. In a satellite communication system where the transmission rate is high and the round-trip delay is long, continuous ARQ is normally used. Stop-and-wait ARQ is used in systems where the time taken to transmit a code word is long compared to the time taken to receive an acknowledgment. Stop-and-wait ARQ is designed for use on half-duplex channels, whereas continuous ARQ is designed for use on full-duplex channels.

The major advantage of ARQ over FEC is that error detection requires much simpler decoding equipment than does error correction. Also, ARQ is adaptive in the sense that information is retransmitted only when errors occur. On the other hand, when the channel error rate is high, retransmissions must be sent too frequently, and the system throughput, the rate at which newly generated messages are correctly received, is lowered by ARQ. In this situation, a combination of FEC for the most frequent error patterns, together with error detection and retransmission for the less likely error patterns, is more efficient than ARQ alone. Although this hybrid ARQ error control strategy has not been implemented in many systems, it clearly carries

the potential for improving throughput in two-way systems subject to a high channel error rate. Various types of ARQ and hybrid ARQ schemes are discussed in Chapter 15 and Section 17.5.

REFERENCES

1. C. E. Shannon, "A Mathematical Theory of Communication," *Bell Syst. Tech. J.*, 27, pp. 379–423 (Part I), 623–656 (Part II), July 1948.
2. T. Berger, *Rate Distortion Theory*, Prentice-Hall, Englewood Cliffs, N.J., 1971.
3. L. Davisson and R. Gray, eds., *Data Compression*, Dowden, Hutchinson, & Ross, Stroudsburg, Pa., 1976.
4. J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering*, Wiley, New York, 1965.
5. A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*, McGraw-Hill, New York, 1979.
6. R. G. Gallager, *Information Theory and Reliable Communication*, Wiley, New York, 1968.

2

Introduction to Algebra

The purpose of this chapter is to provide the reader with an elementary knowledge of algebra that will aid in the understanding of the material in the following chapters. The treatment is basically descriptive and no attempt is made to be mathematically rigorous. There are many good textbooks on algebra. The reader who is interested in more advance algebraic coding theory is referred to the textbooks listed at the end of the chapter. Birkhoff and MacLane [2] is probably the most easily understood text on modern algebra. Fraleigh [4] is also a good and fairly simple text.

2.1 GROUPS

Let G be a set of elements. A *binary operation* $*$ on G is a *rule* that assigns to each pair of elements a and b a uniquely defined third element $c = a * b$ in G. When such a binary operation $*$ is defined on G, we say that G is *closed* under $*$. For example, let G be the set of all integers and let the binary operation on G be real addition $+$. We all know that, for any two integers i and j in G, $i + j$ is a uniquely defined integer in G. Hence, the set of integers is closed under real addition. A binary operation $*$ on G is said to be *associative* if, for any a, b, and c in G,

$$a * (b * c) = (a * b) * c.$$

Now, we introduce a useful algebraic system called a *group*.

Definition 2.1. A set G on which a binary operation $*$ is defined is called a *group* if the following conditions are satisfied:

(i) The binary operation $*$ is associative.

(ii) G contains an element e such that, for any a in G,

$$a * e = e * a = a.$$

This element e is called an *identity* element of G.

(iii) For any element a in G, there exists another element a' in G such that

$$a * a' = a' * a = e.$$

The element a' is called an *inverse* of a (a is also an inverse of a').

A group G is said to be *commutative* if its binary operation $*$ also satisfies the following condition: For any a and b in G,

$$a * b = b * a.$$

Theorem 2.1. The identity element in a group G is unique.

Proof. Suppose that there exist two identity elements e and e' in G. Then $e' = e' * e = e$. This implies that e and e' are identical. Therefore, there is one and only one identity element. Q.E.D.

Theorem 2.2. The inverse of a group element is unique.

Proof. Suppose that there exist two inverses a' and a'' for a group element a. Then

$$a' = a' * e = a' * (a * a'') = (a' * a) * a'' = e * a'' = a''.$$

This implies that a' and a'' are identical and there is only one inverse for a. Q.E.D.

The set of all integers is a commutative group under real addition. In this case, the integer 0 is the identity element and the integer $-i$ is the inverse of integer i. The set of all rational numbers excluding zero is a commutative group under real multiplication. The integer 1 is the identity element with respect to real multiplication, and the rational number b/a is the multiplicative inverse of a/b. The groups noted above contain infinite numbers of elements. Groups with finite numbers of elements do exist, as we shall see in the next example.

Example 2.1

Consider the set of two integers, $G = \{0, 1\}$. Let us define a binary operation, denoted by $\oplus$, on G as follows:

$$0 \oplus 0 = 0, \qquad 0 \oplus 1 = 1, \qquad 1 \oplus 0 = 1, \qquad 1 \oplus 1 = 0.$$

This binary operation is called *modulo-2* addition. The set $G = \{0, 1\}$ is a group under modulo-2 addition. It follows from the definition of modulo-2 addition $\oplus$ that G is closed under $\oplus$ and $\oplus$ is commutative. We can easily check that $\oplus$ is associative. The element 0 is the identity element. The inverse of 0 is itself and the inverse of 1 is also itself. Thus, G together with $\oplus$ is a commutative group.

The number of elements in a group is called the *order* of the group. A group of finite order is called a *finite* group. For any positive integer m, it is possible to construct

a group of order m under a binary operation which is very similar to real addition. This is shown in the next example.

Example 2.2

Let m be a positive integer. Consider the set of integers $G = \{0, 1, 2, \ldots, m-1\}$. Let $+$ denote real addition. Define a binary operation $\boxplus$ on G as follows: For any integers i and j in G,

$$i \boxplus j = r,$$

where r is the *remainder* resulting from dividing $i + j$ by m. The remainder r is an integer between 0 and $m - 1$ (Euclid's division algorithm) and is therefore in G. Hence, G is closed under the binary operation $\boxplus$, which is called *modulo-m addition*. The set $G = \{0, 1, \ldots, m-1\}$ is a group under modulo-m addition. First we see that 0 is the identity element. For $0 < i < m$, i and $m - i$ are both in G. Since

$$i + (m - i) = (m - i) + i = m,$$

it follows from the definition of modulo-m addition that

$$i \boxplus (m - i) = (m - i) \boxplus i = 0.$$

Therefore, i and $m - i$ are inverses to each other with respect to $\boxplus$. It is also clear that the inverse of 0 is itself. Since real addition is commutative, it follows from the definition of modulo-m addition that, for any i and j in G, $i \boxplus j = j \boxplus i$. Therefore, modulo-$m$ addition is commutative. Next, we show that modulo-m addition is also associative. Let i, j, and k be three integers in G. Since real addition is associative, we have

$$i + j + k = (i + j) + k = i + (j + k).$$

Dividing $i + j + k$ by m, we obtain

$$i + j + k = qm + r,$$

where q and r are the quotient and the remainder, respectively, and $0 \leq r < m$. Now, dividing $i + j$ by m, we have

$$i + j = q_1 m + r_1 \tag{2.1}$$

with $0 \leq r_1 < m$. Therefore, $i \boxplus j = r_1$. Dividing $r_1 + k$ by m, we obtain

$$r_1 + k = q_2 m + r_2 \tag{2.2}$$

with $0 \leq r_2 < m$. Hence, $r_1 \boxplus k = r_2$ and

$$(i \boxplus j) \boxplus k = r_2.$$

Combining (2.1) and (2.2), we have

$$i + j + k = (q_1 + q_2)m + r_2.$$

This implies that r_2 is also the remainder when $i + j + k$ is divided by m. Since the remainder resulting from dividing an integer by another integer is unique, we must have $r_2 = r$. As a result, we have

$$(i \boxplus j) \boxplus k = r.$$

Similarly, we can show that

$$i \boxplus (j \boxplus k) = r.$$

Therefore, $(i \boxplus j) \boxplus k = i \boxplus (j \boxplus k)$ and modulo-m addition is associative. This concludes our proof that the set $G = \{0, 1, 2, \ldots, m-1\}$ is a group under modulo-

m addition. We shall call this group an *additive* group. For $m = 2$, we obtain the binary group given in Example 2.1.

The additive group under modulo-5 addition is given by Table 2.1.

TABLE 2.1 MODULO-5 ADDITION

$\boxplus$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

Finite groups with a binary operation similar to real multiplication can also be constructed.

Example 2.3

Let p be a prime (e.g., $p = 2, 3, 5, 7, 11, \ldots$). Consider the set of integers, $G = \{1, 2, 3, \ldots, p-1\}$. Let $\cdot$ denote real multiplication. Define a binary operation $\boxdot$ on G as follows: For i and j in G,

$$i \boxdot j = r,$$

where r is the remainder resulting from dividing $i \cdot j$ by p. First we note that $i \cdot j$ is not divisible by p. Hence, $0 < r < p$ and r is an element in G. Therefore, the set G is closed under the binary operation $\boxdot$, which is referred to as *modulo-p multiplication*. The set $G = \{1, 2, \ldots, p-1\}$ is a group under modulo-p multiplication. We can easily check that modulo-p multiplication is commutative and associative. The identity element is 1. The only thing left to be proved is that every element in G has an inverse. Let i be an element in G. Since p is a prime and $i < p$, i and p must be relatively prime (i.e., i and p do not have any common factor greater than 1). It is well known that there exist two integers a and b such that

$$a \cdot i + b \cdot p = 1 \tag{2.3}$$

and a and p are relatively prime (Euclid's theorem). Rearranging (2.3), we have

$$a \cdot i = -b \cdot p + 1. \tag{2.4}$$

This says that when $a \cdot i$ is divided by p, the remainder is 1. If $0 < a < p$, a is in G and it follows from (2.4) and the definition of modulo-p multiplication that

$$a \boxdot i = i \boxdot a = 1.$$

Therefore, a is the inverse of i. However, if a is not in G, we divide a by p,

$$a = q \cdot p + r. \tag{2.5}$$

Since a and p are relatively prime, the remainder r cannot be 0 and it must be between 1 and $p - 1$. Therefore, r is in G. Now, combining (2.4) and (2.5), we obtain

$$r \cdot i = -(b + qi)p + 1.$$

Therefore, $r \boxdot i = i \boxdot r = 1$ and r is the inverse of i. Hence, any element i in G has an inverse with respect to modulo-p multiplication. The group $G = \{1, 2, \ldots, p-1\}$ under modulo-p multiplication is called a *multiplicative* group. For $p = 2$, we obtain a group $G = \{1\}$ with only one element under modulo-2 multiplication.

If p is *not* a prime, the set $G = \{1, 2, \ldots, p-1\}$ is not a group under modulo-p multiplication (see Problem 2.3). Table 2.2 illustrates the group $G = \{1, 2, 3, 4\}$ under modulo-5 multiplication.

TABLE 2.2 MODULO-5 MULTIPLICATION

$\cdot$	1	2	3	4
1	1	2	3	4
2	2	4	1	3
3	3	1	4	2
4	4	3	2	1

Let H be a nonempty subset of G. The subset H is said to be a *subgroup* of G if H is closed under the group operation of G and satisfies all the conditions of a group. For example, the set of all rational numbers is a group under real addition. The set of all integers is a subgroup of the group of rational numbers under real addition.

2.2 FIELDS

Now, we use the group concepts to introduce another algebraic system, called a *field*. Roughly speaking, a field is a set of elements in which we can do addition, subtraction, multiplication, and division without leaving the set. Addition and multiplication must satisfy the commutative, associative, and distributive laws. A formal definition of a field is given below.

Definition 2.2. Let F be a set of elements on which two binary operations, called addition "$+$" and multiplication "$\cdot$," are defined. The set F together with the two binary operations $+$ and $\cdot$ is a field if the following conditions are satisfied:

(i) F is a commutative group under addition $+$. The identity element with respect to addition is called the *zero* element or the additive identity of F and is denoted by 0.

(ii) The set of nonzero elements in F is a commutative group under multiplication $\cdot$. The identity element with respect to multiplication is called the *unit* element or the multiplicative identity of F and is denoted by 1.

(iii) Multiplication is *distributive* over addition; that is, for any three elements a, b, and c in F,

$$a \cdot (b + c) = a \cdot b + a \cdot c.$$

It follows from the definition that a field consists of at least two elements, the additive identity and the multiplicative identity. Later, we will show that a field of two elements does exist. The number of elements in a field is called the *order* of the

field. A field with finite number of elements is called a *finite field.* In a field, the additive inverse of an element a is denoted by $-a$, and the multiplicative inverse of a is denoted by a^{-1}, provided that $a \neq 0$. Subtracting a field element b from another field element a is defined as adding the additive inverse $-b$ of b to a [i.e., $a - b \triangleq a + (-b)$]. If b is a nonzero element, dividing a by b is defined as multiplying a by the multiplicative inverse b^{-1} of b [i.e., $a \div b \triangleq a \cdot b^{-1}$].

A number of basic properties of fields can be derived from the definition of a field.

Property I. For every element a in a field, $a \cdot 0 = 0 \cdot a = 0$.

Proof. First we note that

$$a = a \cdot 1 = a \cdot (1 + 0) = a + a \cdot 0.$$

Adding $-a$ to both sides of the equality above, we have

$$-a + a = -a + a + a \cdot 0$$
$$0 = 0 + a \cdot 0$$
$$0 = a \cdot 0.$$

Similarly, we can show that $0 \cdot a = 0$. Therefore, we obtain $a \cdot 0 = 0 \cdot a = 0$.

Q.E.D.

Property II. For any two nonzero elements a and b in a field, $a \cdot b \neq 0$.

Proof. This is a direct consequence of the fact that the nonzero elements of a field are closed under multiplication. Q.E.D.

Property III. $a \cdot b = 0$ and $a \neq 0$ imply that $b = 0$.

Proof. This is a direct consequence of Property II. Q.E.D.

Property IV. For any two elements a and b in a field,

$$-(a \cdot b) = (-a) \cdot b = a \cdot (-b).$$

Proof. $0 = 0 \cdot b = (a + (-a)) \cdot b = a \cdot b + (-a) \cdot b$. Therefore, $(-a) \cdot b$ must be the additive inverse of $a \cdot b$ and $-(a \cdot b) = (-a) \cdot b$. Similarly, we can prove that $-(a \cdot b) = a \cdot (-b)$. Q.E.D.

Property V. For $a \neq 0$, $a \cdot b = a \cdot c$ implies that $b = c$.

Proof. Since a is a nonzero element in the field, it has a multiplicative inverse a^{-1}. Multiplying both side of $a \cdot b = a \cdot c$ by a^{-1}, we obtain

$$a^{-1} \cdot (a \cdot b) = a^{-1} \cdot (a \cdot c)$$
$$(a^{-1} \cdot a) \cdot b = (a^{-1} \cdot a) \cdot c$$
$$1 \cdot b = 1 \cdot c.$$

Thus, $b = c$. Q.E.D.

We can verify readily that the set of real numbers is a field under real number addition and multiplication. This field has an infinite number of elements. Fields with

finite number of elements can be constructed and are illustrated in the next two examples and in Section 2.4.

Example 2.4

Consider the set {0, 1} together with modulo-2 addition and multiplication shown in Tables 2.3 and 2.4. In Example 2.1 we have shown that {0, 1} is a commutative group under modulo-2 addition; and in Example 2.3, we have shown that {1} is a group under modulo-2 multiplication. We can easily check that modulo-2 multiplication is distributive over modulo-2 addition by simply computing $a \cdot (b + c)$ and $a \cdot b + a \cdot c$ for eight possible combinations of a, b and c ($a = 0$ or 1, $b = 0$ or 1 and $c = 0$ or 1). Therefore, the set {0, 1} is a field of two elements under modulo-2 addition and modulo-2 multiplication.

TABLE 2.3
MODULO-2 ADDITION

+	0	1
0	0	1
1	1	0

TABLE 2.4
MODULO-2 MULTIPLICATION

·	0	1
0	0	0
1	0	1

The field given in Example 2.4 is usually called a *binary* field and is denoted by GF(2). The binary field GF(2) plays an important role in coding theory and is widely used in digital computers and digital data transmission (or storage) systems.

Example 2.5

Let p be a prime. We have shown in Example 2.2 that the set of integers $\{0, 1, 2, \ldots, p - 1\}$ is a commutative group under modulo-p addition. We have also shown in Example 2.3 that the nonzero elements, $\{1, 2, \ldots, p - 1\}$ form a commutative group under modulo-p multiplication. Following the definitions of modulo-p addition and multiplication and the fact that real number multiplication is distributive over real number addition, we can show that modulo-p multiplication is distributive over modulo-p addition. Therefore, the set $\{0, 1, 2, \ldots, p - 1\}$ is a field of order p under modulo-p addition and multiplication. Since this field is constructed from a prime p, it is called a *prime field* and is denoted by GF(p). For $p = 2$, we obtain the binary field GF(2).

Let $p = 7$. Modulo-7 addition and multiplication are given by Tables 2.5 and 2.6, respectively. The set of integers {0, 1, 2, 3, 4, 5, 6} is a field of seven elements, denoted by GF(7), under modulo-7 addition and multiplication. The addition table is also used for subtraction. For example, if we want to subtract 6 from 3, we first use the addition table to find the additive inverse of 6, which is 1. Then we add 1 to 3 to obtain the result [i.e., $3 - 6 = 3 + (-6) = 3 + 1 = 4$]. For division, we use the multiplication table. Suppose that we divide 3 by 2. We first find the multiplicative inverse of 2, which is 4, and then we multiply 3 by 4 to obtain the result [i.e., $3 \div 2 = 3 \cdot (2^{-1}) = 3 \cdot 4 = 5$]. Here we have demonstrated that, in a finite field, addition, subtraction, multiplication, and division can be carried out in a manner similar to ordinary arithmetic, with which we are quite familiar.

In Example 2.5 we have shown that, for any prime p, there exists a finite field

TABLE 2.5
MODULO-7 ADDITION

+	0	1	2	3	4	5	6
0	0	1	2	3	4	5	6
1	1	2	3	4	5	6	0
2	2	3	4	5	6	0	1
3	3	4	5	6	0	1	2
4	4	5	6	0	1	2	3
5	5	6	0	1	2	3	4
6	6	0	1	2	3	4	5

TABLE 2.6
MODULO-7 MULTIPLICATION

·	0	1	2	3	4	5	6
0	0	0	0	0	0	0	0
1	0	1	2	3	4	5	6
2	0	2	4	6	1	3	5
3	0	3	6	2	5	1	4
4	0	4	1	5	2	6	3
5	0	5	3	1	6	4	2
6	0	6	5	4	3	2	1

of p elements. In fact, for any positive integer m, it is possible to extend the prime field GF(p) to a field of p^m elements which is called an *extension field* of GF(p) and is denoted by GF(p^m). Furthermore, it has been proved that the order of any finite field is a power of a prime. Finite fields are also called *Galois* fields, in honor of their discoverer. A large portion of algebraic coding theory, code construction, and decoding is built around finite fields. In the rest of this section and in the next two sections we examine some basic structures of finite fields, their arithmetic, and the construction of extension fields from prime fields. Our presentation will be mainly descriptive and no attempt is made to be mathematically rigorous. Since finite-field arithmetic is very similar to ordinary arithmetic, most of the rules of ordinary arithmetic apply to finite-field arithmetic. Therefore, it is possible to utilize most of the techniques of algebra in the computations over finite fields.

Consider a finite field of q elements, GF(q). Let us form the following sequence of sums of the unit element 1 in GF(q):

$$\sum_{i=1}^{1} 1 = 1, \quad \sum_{i=1}^{2} 1 = 1 + 1, \quad \sum_{i=1}^{3} 1 = 1 + 1 + 1, \quad \ldots,$$

$$\sum_{i=1}^{k} 1 = 1 + 1 + \cdots + 1 \ (k \text{ times}), \quad \ldots$$

Since the field is closed under addition, these sums must be elements in the field. Since the field has finite number of elements, these sums cannot be all distinct. Therefore, at some point of the sequence of sums, there must be a repetition; that is, there must exist two positive integers m and n such that $m < n$ and

$$\sum_{i=1}^{m} 1 = \sum_{i=1}^{n} 1.$$

This implies that $\sum_{i=1}^{n-m} 1 = 0$. Therefore, there must exist a *smallest positive integer* λ such that $\sum_{i=1}^{\lambda} 1 = 0$. This integer λ is called the *characteristic* of the field GF(q). The characteristic of the binary field GF(2) is 2, since $1 + 1 = 0$. The characteristic of the prime field GF(p) is p, since $\sum_{i=1}^{k} 1 = k \neq 0$ for $1 \leq k < p$ and $\sum_{i=1}^{p} 1 = 0$.

Theorem 2.3. The characteristic λ of a finite field is prime.

Proof. Suppose that λ is not a prime and is equal to the product of two smaller integers k and m (i.e., $\lambda = km$). Since the field is closed under multiplication,

$$\left(\sum_{i=1}^{k} 1\right) \cdot \left(\sum_{i=1}^{m} 1\right)$$

is also a field element. It follows from the distributive law that

$$\left(\sum_{i=1}^{k} 1\right) \cdot \left(\sum_{i=1}^{m} 1\right) = \sum_{i=1}^{km} 1.$$

Since $\sum_{i=1}^{km} 1 = 0$, then either $\sum_{i=1}^{k} 1 = 0$ or $\sum_{i=1}^{m} 1 = 0$. However, this contradicts the definition that λ is the smallest positive integer such that $\sum_{i=1}^{\lambda} 1 = 0$. Therefore, we conclude that λ is prime. Q.E.D.

It follows from the definition of the characteristic of a finite field that for any two distinct positive integers k and m less than λ,

$$\sum_{i=1}^{k} 1 \neq \sum_{i=1}^{m} 1.$$

Suppose that $\sum_{i=1}^{k} 1 = \sum_{i=1}^{m} 1$. Then we have

$$\sum_{i=1}^{m-k} 1 = 0$$

(assuming that $m > k$). However, this is impossible since $m - k < \lambda$. Therefore, the sums

$$1 = \sum_{i=1}^{1} 1, \quad \sum_{i=1}^{2} 1, \quad \sum_{i=1}^{3} 1, \quad \ldots, \quad \sum_{i=1}^{\lambda-1} 1, \quad \sum_{i=1}^{\lambda} 1 = 0$$

are λ distinct elements in GF(q). In fact, this set of sums itself is a field of λ elements, GF(λ), under the addition and multiplication of GF(q) (see Problem 2.6). Since GF(λ) is a subset of GF(q), GF(λ) is called a *subfield* of GF(q). Therefore, any finite field GF(q) of characteristic λ contains a subfield of λ elements. It can be proved that if $q \neq \lambda$, then q is a power of λ.

Now let a be a nonzero element in GF(q). Since the set of nonzero elements of GF(q) is closed under multiplication, the following powers of a,

$$a^1 = a, \quad a^2 = a \cdot a, \quad a^3 = a \cdot a \cdot a, \quad \ldots$$

must also be nonzero elements in GF(q). Since GF(q) has only a finite number of elements, the powers of a given above cannot all be distinct. Therefore, at some point of the sequence of powers of a, there must be a repetition; that is, there must exist two positive integers k and m such that $m > k$ and $a^k = a^m$. Let a^{-1} be the multiplicative inverse of a. Then $(a^{-1})^k = a^{-k}$ is the multiplicative inverse of a^k. Multiplying both sides of $a^k = a^m$ by a^{-k}, we obtain

$$1 = a^{m-k}.$$

This implies that there must exist a *smallest positive integer* n such that $a^n = 1$. This integer n is called the *order* of the field element a. Therefore, the sequence $a^1, a^2, a^3, \ldots$ repeats itself after $a^n = 1$. Also, the powers $a^1, a^2, \ldots, a^{n-1}, a^n = 1$ are all distinct. In fact, they form a group under the multiplication of GF(q). First we see that they contain the unit element 1. Consider $a^i \cdot a^j$. If $i + j \leq n$,

$$a^i \cdot a^j = a^{i+j}.$$

If $i + j > n$, we have $i + j = n + r$, where $0 < r \leq n$. Hence,

$$a^i \cdot a^j = a^{i+j} = a^n \cdot a^r = a^r.$$

Therefore, the powers $a^1, a^2, \ldots, a^{n-1}, a^n = 1$ are closed under the multiplication of GF(q). For $1 \leq i < n$, a^{n-i} is the multiplicative inverse of a^i. Since the powers of a are nonzero elements in GF(q), they satisfy the associative and commutative laws. Therefore, we conclude that $a^n = 1, a^1, a^2, \ldots, a^{n-1}$ form a group under the multiplication of GF(q). A group is said to be *cyclic* if there exists an element in the group whose powers constitute the whole group.

Theorem 2.4. Let a be a nonzero element of a finite field GF(q). Then $a^{q-1} = 1$.

Proof. Let $b_1, b_2, \ldots, b_{q-1}$ be the $q - 1$ nonzero elements of GF(q). Clearly, the $q - 1$ elements, $a \cdot b_1, a \cdot b_2, \ldots, a \cdot b_{q-1}$, are nonzero and distinct. Thus,

$$(a \cdot b_1) \cdot (a \cdot b_2) \cdots (a \cdot b_{q-1}) = b_1 \cdot b_2 \cdots b_{q-1}$$

$$a^{q-1} \cdot (b_1 \cdot b_2 \cdots b_{q-1}) = b_1 \cdot b_2 \cdots b_{q-1}.$$

Since $a \neq 0$ and $(b_1 \cdot b_2 \cdots b_{q-1}) \neq 0$, we must have $a^{q-1} = 1$. Q.E.D.

Theorem 2.5. Let a be a nonzero element in a finite field GF(q). Let n be the order of a. Then n divides $q - 1$.

Proof. Suppose that n does not divide $q - 1$. Dividing $q - 1$ by n, we obtain

$$q - 1 = kn + r,$$

where $0 < r < n$. Then

$$a^{q-1} = a^{kn+r} = a^{kn} \cdot a^r = (a^n)^k \cdot a^r.$$

Since $a^{q-1} = 1$ and $a^n = 1$, we must have $a^r = 1$. This is impossible since $0 < r < n$ and n is the smallest integer such that $a^n = 1$. Therefore, n must divide $q - 1$. Q.E.D.

In a finite field GF(q), a nonzero element a is said to be *primitive* if the order of a is $q - 1$. Therefore, the powers of a primitive element generate all the nonzero elements of GF(q). Every finite field has a primitive element (see Problem 2.7).

Consider the prime field GF(7) illustrated by Tables 2.5 and 2.6. The characteristic of this field is 7. If we take the powers of the integer 3 in GF(7) using the multiplication table, we obtain

$$3^1 = 3, \quad 3^2 = 3 \cdot 3 = 2, \quad 3^3 = 3 \cdot 3^2 = 6,$$

$$3^4 = 3 \cdot 3^3 = 4, \quad 3^5 = 3 \cdot 3^4 = 5, \quad 3^6 = 3 \cdot 3^5 = 1.$$

Therefore, the order of the integer 3 is 6 and the integer 3 is a primitive element of GF(7). The powers of the integer 4 is GF(7) are

$$4^1 = 4, \quad 4^2 = 4 \cdot 4 = 2, \quad 4^3 = 4 \cdot 4^2 = 1.$$

Clearly, the order of the integer 4 is 3, which is a factor of 6.

2.3 BINARY FIELD ARITHMETIC

In general, we can construct code with symbols from any Galois field GF(q), where q is either a prime p or a power of p. However, codes with symbols from the binary field GF(2) or its extension GF(2^m) are most widely used in digital data transmission and storage systems because information in these systems is universally coded in binary form for practical reasons. In this book we are concerned only with binary codes and codes with symbols from the field GF(2^m). Most of the results presented in this book can be generalized to codes with symbols from any finite field GF(q) with $q \neq 2$ or 2^m. In this section we discuss arithmetic over the binary field GF(2), which will be used in the rest of this book.

In binary arithmetic we use modulo-2 addition and multiplication, which are defined by Tables 2.3 and 2.4, respectively. This arithmetic is actually equivalent to ordinary arithmetic, except that we consider 2 to be equal to 0 (i.e., $1 + 1 = 2 = 0$). Note that since $1 + 1 = 0$, $1 = -1$. Hence, in binary arithmetic, subtraction is the same as addition. To illustrate how the ideas of ordinary algebra can be used with the binary arithmetic, we consider the following sets of equations:

$$X + Y = 1$$

$$X + Z = 0$$

$$X + Y + Z = 1.$$

These can be solved by adding the first equation to the third, giving $Z = 0$. Then from the second equation, since $Z = 0$ and $X + Z = 0$, we obtain $X = 0$. From the first equation, since $X = 0$ and $X + Y = 1$, we have $Y = 1$. We can substitute these solutions back into the original set of equations and verify that they are correct.

Since we were able to solve the equations shown above, they must be linearly independent, and the determinant of the coefficients on the left side must be nonzero. If the determinant is nonzero, it must be 1. This can be verified as follows:

$$\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix} = 1 \cdot \begin{vmatrix} 0 & 1 \\ 1 & 1 \end{vmatrix} - 1 \cdot \begin{vmatrix} 1 & 1 \\ 1 & 1 \end{vmatrix} + 0 \cdot \begin{vmatrix} 1 & 0 \\ 1 & 1 \end{vmatrix}$$

$$= 1 \cdot 1 - 1 \cdot 0 + 0 \cdot 1 = 1.$$

We could have solved the equations by Cramer's rule:

$$X = \frac{\begin{vmatrix} 1 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{0}{1} = 0, \qquad Y = \frac{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{1}{1} = 1,$$

$$Z = \frac{\begin{vmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{vmatrix}} = \frac{0}{1} = 0.$$

Next we consider computations with polynomials whose coefficients are from the binary field GF(2). A polynomial $f(X)$ with one *variable* X and with coefficients from GF(2) is of the following form:

$$f(X) = f_0 + f_1X + f_2X^2 + \cdots + f_nX^n,$$

where $f_i = 0$ or 1 for $0 \leq i \leq n$. The *degree* of a polynomial is the largest power of X with a nonzero coefficient. For the polynomial above, if $f_n = 1$, $f(X)$ is a polynomial of degree n; if $f_n = 0$, $f(X)$ is a polynomial of degree less than n. The degree of $f(X) = f_0$ is zero. In the following we use the phrase "a polynomial over GF(2)" to mean "a polynomial with coefficients from GF(2)." There are two polynomials over GF(2) with degree 1: X and $1 + X$. There are four polynomials over GF(2) with degree 2: X^2, $1 + X^2$, $X + X^2$, and $1 + X + X^2$. In general, there are 2^n polynomials over GF(2) with degree n.

Polynomials over GF(2) can be added (or subtracted), multiplied, and divided in the usual way. Let

$$g(X) = g_0 + g_1X + g_2X^2 + \cdots + g_mX^m$$

be another polynomial over GF(2). To add $f(X)$ and $g(X)$, we simply add the coefficients of the same power of X in $f(X)$ and $g(X)$ as follows (assuming that $m \leq n$):

$$f(X) + g(X) = (f_0 + g_0) + (f_1 + g_1)X + \cdots + (f_m + g_m)X^m + f_{m+1}X^{m+1} + \cdots + f_nX^n,$$

where $f_i + g_i$ is carried out in modulo-2 addition. For example, adding $a(X) = 1 + X + X^3 + X^5$ and $b(X) = 1 + X^2 + X^3 + X^4 + X^7$, we obtain the following sum:

$$\begin{aligned} a(X) + b(X) &= (1 + 1) + X + X^2 + (1 + 1)X^3 + X^4 + X^5 + X^7 \\ &= X + X^2 + X^4 + X^5 + X^7. \end{aligned}$$

When we multiply $f(X)$ and $g(X)$, we obtain the following product:

$$f(X) \cdot g(X) = c_0 + c_1X + c_2X^2 + \cdots + c_{n+m}X^{n+m},$$

where

$$\begin{aligned} c_0 &= f_0g_0 \\ c_1 &= f_0g_1 + f_1g_0 \\ c_2 &= f_0g_2 + f_1g_1 + f_2g_0 \\ &\vdots \\ c_i &= f_0g_i + f_1g_{i-1} + f_2g_{i-2} + \cdots + f_ig_0 \\ &\vdots \\ c_{n+m} &= f_ng_m. \end{aligned} \tag{2.6}$$

(Multiplication and addition of coefficients are modulo-2.) It is clear from (2.6) that if $g(X) = 0$, then

$$f(X) \cdot 0 = 0. \tag{2.7}$$

We can readily verify that the polynomials over GF(2) satisfy the following conditions:

(i) Commutative:

$$\begin{aligned} a(X) + b(X) &= b(X) + a(X) \\ a(X) \cdot b(X) &= b(X) \cdot a(X). \end{aligned}$$

(ii) Associative:

$$\begin{aligned} a(X) + [b(X) + c(X)] &= [a(X) + b(X)] + c(X) \\ a(X) \cdot [b(X) \cdot c(X)] &= [a(X) \cdot b(X)] \cdot c(X). \end{aligned}$$

(iii) Distributive:

$$a(X) \cdot [b(X) + c(X)] = [a(X) \cdot b(X)] + [a(X) \cdot c(X)]. \tag{2.8}$$

Suppose that the degree of $g(X)$ is *not* zero. When $f(X)$ is divided by $g(X)$, we obtain a unique pair of polynomials over GF(2)—$q(X)$, called the quotient, and $r(X)$, called the remainder—such that

$$f(X) = q(X)g(X) + r(X)$$

and the degree of $r(X)$ is less than that of $g(X)$. This is known as Euclid's division algorithm. As an example, we divide $f(X) = 1 + X + X^4 + X^5 + X^6$ by $g(X) = 1 + X + X^3$. Using the long-division technique, we have

$$\begin{array}{r} X^3 + X^2 \quad \text{(quotient)} \\ \hline X^3 + X + 1 \;\big)\; X^6 + X^5 + X^4 \qquad\qquad + X + 1 \\ \underline{X^6 \qquad + X^4 + X^3} \\ \qquad X^5 \qquad + X^3 \qquad + X + 1 \\ \underline{X^5 \qquad + X^3 + X^2} \\ \qquad\qquad\qquad X^2 + X + 1 \quad \text{(remainder).} \end{array}$$

We can easily verify that

$$X^6 + X^5 + X^4 + X + 1 = (X^3 + X^2)(X^3 + X + 1) + X^2 + X + 1.$$

When $f(X)$ is divided by $g(X)$, if the remainder $r(X)$ is identical to zero [$r(X) = 0$], we say that $f(X)$ is divisible by $g(X)$ and $g(X)$ is a factor of $f(X)$.

For real numbers, if a is a *root* of a polynomial $f(X)$ [i.e., $f(a) = 0$], $f(X)$ is divisible by $x - a$. (This fact follows from Euclid's division algorithm.) This is still true for $f(X)$ over GF(2). For example, let $f(X) = 1 + X^2 + X^3 + X^4$. Substituting $X = 1$, we obtain

$$f(1) = 1 + 1^2 + 1^3 + 1^4 = 1 + 1 + 1 + 1 = 0.$$

Thus, $f(X)$ has 1 as a root and it should be divisible by $X + 1$.

$$\begin{array}{r} X^3 + X + 1 \\ \hline X + 1 \;\big)\; X^4 + X^3 + X^2 \qquad + 1 \\ \underline{X^4 + X^3} \\ \qquad\qquad X^2 \qquad + 1 \\ \qquad\qquad \underline{X^2 + X} \\ \qquad\qquad\qquad X + 1 \\ \qquad\qquad\qquad \underline{X + 1} \\ \qquad\qquad\qquad\quad 0 \end{array}$$

For a polynomial $f(X)$ over GF(2), if it has an even number of terms, it is divisible by $X+1$. A polynomial $p(X)$ over GF(2) of degree m is said to be *irreducible* over GF(2) if $p(X)$ is not divisible by any polynomial over GF(2) of degree less than m but greater than zero. Among the four polynomials of degree 2, X^2, X^2+1 and X^2+X are not irreducible since they are either divisible by X or $X+1$. However, X^2+X+1 does not have either "0" or "1" as a root and so is not divisible by any polynomial of degree 1. Therefore, X^2+X+1 is an irreducible polynomial of degree 2. The polynomial X^3+X+1 is an irreducible polynomial of degree 3. First we note that X^3+X+1 does not have either 0 or 1 as a root. Therefore, X^3+X+1 is not divisible by X or $X+1$. Since it is not divisible by any polynomial of degree 1, it cannot be divisible by a polynomial of degree 2. Consequently, X^3+X+1 is irreducible over GF(2). We may verify that X^4+X+1 is an irreducible polynomial of degree 4. It has been proved that, for any $m \geq 1$, there exists an irreducible polynomial degree m. An important theorem regarding irreducible polynomials over GF(2) is given below without a proof.

Theorem 2.6. Any irreducible polynomial over GF(2) of degree m divides $X^{2^m-1}+1$.

As an example of Theorem 2.6, we can check that X^3+X+1 divides $X^{2^3-1}+1=X^7+1$:

$$
\begin{array}{r}
X^4+X^2+X+1 \\
\hline
X^3+X+1 \;)\; X^7 \qquad\qquad\qquad\qquad\quad +1 \\
\underline{X^7 \quad +X^5+X^4} \\
X^5+X^4 \qquad\qquad\quad +1 \\
\underline{X^5 \qquad +X^3+X^2} \\
X^4+X^3+X^2 \quad +1 \\
\underline{X^4 \qquad +X^2+X} \\
X^3 \qquad +X+1 \\
\underline{X^3 \qquad +X+1} \\
0.
\end{array}
$$

An irreducible polynomial $p(X)$ of degree m is said to be *primitive* if the smallest positive integer n for which $p(X)$ divides X^n+1 is $n=2^m-1$. We may check that $p(X)=X^4+X+1$ divides $X^{15}+1$ but does not divide any X^n+1 for $1 \leq n < 15$. Hence, X^4+X+1 is a primitive polynomial. The polynomial $X^4+X^3+X^2+X+1$ is irreducible but it is not primitive, since it divides X^5+1. It is not easy to recognize a primitive polynomial. However, there are tables of irreducible polynomials in which primitive polynomials are indicated [6,7]. For a given m, there may be more than one primitive polynomials of degree m. A list of primitive polynomials is given in Table 2.7. For each degree m, we list only a primitive polynomial with the smallest number of terms.

Before leaving this section, we derive another useful property of polynomials over GF(2). Consider

TABLE 2.7 LIST OF PRIMITIVE POLYNOMIALS

m		m	
3	$1+X+X^3$	14	$1+X+X^6+X^{10}+X^{14}$
4	$1+X+X^4$	15	$1+X+X^{15}$
5	$1+X^2+X^5$	16	$1+X+X^3+X^{12}+X^{16}$
6	$1+X+X^6$	17	$1+X^3+X^{17}$
7	$1+X^3+X^7$	18	$1+X^7+X^{18}$
8	$1+X^2+X^3+X^4+X^8$	19	$1+X+X^2+X^5+X^{19}$
9	$1+X^4+X^9$	20	$1+X^3+X^{20}$
10	$1+X^3+X^{10}$	21	$1+X^2+X^{21}$
11	$1+X^2+X^{11}$	22	$1+X+X^{22}$
12	$1+X+X^4+X^6+X^{12}$	23	$1+X^5+X^{23}$
13	$1+X+X^3+X^4+X^{13}$	24	$1+X+X^2+X^7+X^{24}$

$$
\begin{aligned}
f^2(X) &= (f_0+f_1X+\cdots+f_nX^n)^2 \\
&= [f_0+(f_1X+f_2X^2+\cdots+f_nX^n)]^2 \\
&= f_0^2+f_0\cdot(f_1X+f_2X^2+\cdots+f_nX^2) \\
&\qquad +f_0\cdot(f_1X+f_2X^2+\cdots+f_nX^2)+(f_1X+f_2X^2+\cdots+f_nX^n)^2 \\
&= f_0^2+(f_1X+f_2X^2+\cdots+f_nX^n)^2.
\end{aligned}
$$

Expanding the equation above repeatedly, we eventually obtain

$$f^2(X)=f_0^2+(f_1X)^2+(f_2X^2)^2+\cdots+(f_nX^n)^2.$$

Since $f_i=0$ or 1, $f_i^2=f_i$. Hence, we have

$$
\begin{aligned}
f^2(X) &= f_0+f_1X^2+f_2(X^2)^2+\cdots+f_n(X^2)^n \\
&= f(X^2).
\end{aligned} \tag{2.9}
$$

It follows from (2.9) that, for any $l \geq 0$,

$$[f(X)]^{2^l}=f(X^{2^l}). \tag{2.10}$$

2.4 CONSTRUCTION OF GALOIS FIELD GF(2^m)

In this section we present a method for constructing the Galois field of 2^m elements ($m>1$) from the binary field GF(2). We begin with the two elements 0 and 1, from GF(2) and a new symbol α. Then we define a multiplication "$\cdot$" to introduce a sequence of powers of α as follows:

$$
\begin{aligned}
&0\cdot 0=0, \\
&0\cdot 1=1\cdot 0=0, \\
&1\cdot 1=1, \\
&0\cdot\alpha=\alpha\cdot 0=0, \\
&1\cdot\alpha=\alpha\cdot 1=\alpha,
\end{aligned} \tag{2.11}
$$

$$\begin{aligned}
&\alpha^2 = \alpha \cdot \alpha,\\
&\alpha^3 = \alpha \cdot \alpha \cdot \alpha,\\
&\quad\vdots\\
&\alpha^j = \alpha \cdot \alpha \cdot \cdots \cdot \alpha \quad (j \text{ times}),\\
&\quad\vdots
\end{aligned} \tag{2.11}$$

It follows from the definition of multiplication above that

$$\begin{aligned}
&0 \cdot \alpha^j = \alpha^j \cdot 0 = 0,\\
&1 \cdot \alpha^j = \alpha^j \cdot 1 = \alpha^j,\\
&\alpha^i \cdot \alpha^j = \alpha^j \cdot \alpha^i = \alpha^{i+j}.
\end{aligned} \tag{2.12}$$

Now, we have the following set of elements on which a multiplication operation "$\cdot$" is defined:

$$F = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^j, \ldots\}.$$

The element 1 is sometimes denoted α^0.

Next we put a condition on the element α so that the set F contains only 2^m elements and is closed under the multiplication "$\cdot$" defined by (2.11). Let $p(X)$ be a primitive polynomial of degree m over GF(2). We assume that $p(\alpha) = 0$. Since $p(X)$ divides $X^{2^m-1} + 1$ (Theorem 2.6), we have

$$X^{2^m-1} + 1 = q(X)p(X). \tag{2.13}$$

If we replace X by α in (2.13), we obtain

$$\alpha^{2^m-1} + 1 = q(\alpha)p(\alpha).$$

Since $p(\alpha) = 0$, we have

$$\alpha^{2^m-1} + 1 = q(\alpha) \cdot 0.$$

If we regard $q(\alpha)$ as a polynomial of α over GF(2), it follows from (2.7) that $q(\alpha) \cdot 0 = 0$. As a result, we obtain the following equality:

$$\alpha^{2^m-1} + 1 = 0.$$

Adding 1 to both sides of $\alpha^{2^m-1} + 1 = 0$ (use modulo-2 addition) results in the following equality:

$$\alpha^{2^m-1} = 1. \tag{2.14}$$

Therefore, under the condition that $p(\alpha) = 0$, the set F becomes finite and contains the following elements:

$$F^* = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2}\}.$$

The nonzero elements of F^* are closed under the multiplication operation "$\cdot$" defined by (2.11). To see this, let i and j be two integers such that $0 \leq i, j < 2^m - 1$. If $i + j < 2^m - 1$, then $\alpha^i \cdot \alpha^j = \alpha^{i+j}$, which is obviously a nonzero element in F^*. If $i + j \geq 2^m - 1$, we can express $i + j$ as follows: $i + j = (2^m - 1) + r$, where $0 \leq r < 2^m - 1$.

Then

$$\alpha^i \cdot \alpha^j = \alpha^{i+j} = \alpha^{(2^m-1)+r} = \alpha^{2^m-1} \cdot \alpha^r = 1 \cdot \alpha^r = \alpha^r,$$

which is also a nonzero element in F^*. Hence, we conclude that the nonzero elements of F^* are closed under the multiplication "$\cdot$" defined by (2.11). In fact, these nonzero elements form a commutative group under "$\cdot$". First, we see that the element 1 is the unit element. From (2.11) and (2.12) we see readily that the multiplication operation "$\cdot$" is commutative and associative. For $0 < i < 2^m - 1$, α^{2^m-i-1} is the multiplicative inverse of α^i since

$$\alpha^{2^m-i-1} \cdot \alpha^i = \alpha^{2^m-1} = 1.$$

(Note that $\alpha^0 = \alpha^{2^m-1} = 1$.) It will be clear in what follows that $1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2}$ represent $2^m - 1$ distinct elements. Therefore, the nonzero elements of F^* form a group of order $2^m - 1$ under the multiplication operation "$\cdot$" defined by (2.11).

Our next step is to define an addition operation "$+$" on F^* so that F^* forms a commutative group under "$+$." For $0 \leq i < 2^m - 1$, we divide the polynomial X^i by $p(X)$ and obtain the following:

$$X^i = q_i(X)p(X) + a_i(X), \tag{2.15}$$

where $q_i(X)$ and $a_i(X)$ are the quotient and the remainder, respectively. The remainder $a_i(X)$ is a polynomial of degree $m - 1$ or less over GF(2) and is of the following form:

$$a_i(X) = a_{i0} + a_{i1}X + a_{i2}X^2 + \cdots + a_{i,m-1}X^{m-1}.$$

Since X and $p(X)$ are relatively prime (i.e., they do not have any common factor except 1), X^i is not divisible by $p(X)$. Therefore, for any $i \geq 0$,

$$a_i(X) \neq 0. \tag{2.16}$$

For $0 \leq i, j < 2^m - 1$, and $i \neq j$, we can also show that

$$a_i(X) \neq a_j(X). \tag{2.17}$$

Suppose that $a_i(X) = a_j(X)$. Then it follows from (2.15) that

$$\begin{aligned}
X^i + X^j &= [q_i(X) + q_j(X)]p(X) + a_i(X) + a_j(X)\\
&= [q_i(X) + q_j(X)]p(X).
\end{aligned}$$

This implies that $p(X)$ divides $X^i + X^j = X^i(1 + X^{j-i})$ (assuming that $j > i$). Since X^i and $p(X)$ are relatively prime, $p(X)$ must divide $X^{j-i} + 1$. However, this is impossible since $j - i < 2^m - 1$ and $p(X)$ is a primitive polynomial of degree m which does not divide $X^n + 1$ for $n < 2^m - 1$. Therefore, our hypothesis that $a_i(X) = a_j(X)$ is invalid. As a result, for $0 \leq i, j < 2^m - 1$ and $i \neq j$, we must have $a_i(X) \neq a_j(X)$. Hence, for $i = 0, 1, 2, \ldots, 2^m - 2$, we obtain $2^m - 1$ distinct nonzero polynomials $a_i(X)$ of degree $m - 1$ or less. Now, replacing X by α in (2.15) and using the equality that $q_i(\alpha) \cdot 0 = 0$ [see (2.7)], we obtain the following polynomial expression for α^i:

$$\alpha^i = a_i(\alpha) = a_{i0} + a_{i1}\alpha + a_{i2}\alpha^2 + \cdots + a_{i,m-1}\alpha^{m-1}. \tag{2.18}$$

From (2.16), (2.17), and (2.18), we see that the $2^m - 1$ nonzero elements, $\alpha^0, \alpha^1, \ldots, \alpha^{2^m-2}$ in F^*, are represented by $2^m - 1$ *distinct nonzero polynomials* of α over GF(2) with degree $m - 1$ or less. The zero element 0 in F^* may be represented by the *zero*

polynomial. As a result, the 2^m elements in F^* are represented by 2^m *distinct polynomials* of α over GF(2) with degree $m-1$ or less and are regarded as 2^m distinct elements.

Now, we define an addition "+" on F^* as follows:

$$0 + 0 = 0 \tag{2.19a}$$

and, for $0 \leq i, j < 2^m - 1$,

$$0 + \alpha^i = \alpha^i + 0 = \alpha^i, \tag{2.19b}$$

$$\begin{aligned}\alpha^i + \alpha^j &= (a_{i0} + a_{i1}\alpha + \cdots + a_{i,m-1}\alpha^{m-1}) + (a_{j0} + a_{j1}\alpha + \cdots + a_{j,m-1}\alpha^{m-1}) \\ &= (a_{i0} + a_{j0}) + (a_{i1} + a_{j1})\alpha + \cdots + (a_{i,m-1} + a_{j,m-1})\alpha^{m-1},\end{aligned} \tag{2.19c}$$

where $a_{i,l} + a_{j,l}$ is carried out in modulo-2 addition. From (2.19c) we see that, for $i = j$,

$$\alpha^i + \alpha^i = 0 \tag{2.20}$$

and for $i \neq j$,

$$(a_{i0} + a_{j0}) + (a_{i1} + a_{j1})\alpha + \cdots + (a_{i,m-1} + a_{j,m-1})\alpha^{m-1}$$

is nonzero and must be the polynomial expression for some α^k in F^*. Hence, the set F^* is closed under the addition "+" defined by (2.19). We can immediately verify that F^* is a commutative group under "+." First, we see that 0 is the additive identity. Using the fact that modulo-2 addition is commutative and associative, the addition defined on F^* is also commutative and associative. From (2.19a) and (2.20) we see that the additive inverse of any element in F^* is itself.

Up to this point, we have shown that the set $F^* = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-2}\}$ is a commutative group under an addition operation "+" and the nonzero elements of F^* form a commutative group under a multiplication operation "·." Using the polynomial representation for the elements in F^* and (2.8) (polynomial multiplication satisfies distributive law), we readily see that the multiplication on F^* is distributive over the addition on F^*. Therefore, the set $F^* = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^m-1}\}$ is a Galois field of 2^m elements, GF(2^m). We notice that the addition and multiplication defined on $F^* = \text{GF}(2^m)$ imply modulo-2 addition and multiplication. Hence, the subset $\{0, 1\}$ forms a subfield of GF(2^m) [i.e., GF(2) is a subfield of GF(2^m)]. The binary field GF(2) is usually called the *ground* field of GF(2^m). The characteristic of GF(2^m) is 2.

In our process of constructing GF(2^m) from GF(2), we have developed two representations for the nonzero elements of GF(2^m): the power representation and the polynomial representation. The power representation is convenient for multiplication and the polynomial representation is convenient for addition.

Example 2.6

Let $m = 4$. The polynomial $p(X) = 1 + X + X^4$ is a primitive polynomial over GF(2). Set $p(\alpha) = 1 + \alpha + \alpha^4 = 0$. Then $\alpha^4 = 1 + \alpha$. Using this, we can construct GF(2^4). The elements of GF(2^4) are given in Table 2.8. The identity $\alpha^4 = 1 + \alpha$ is used repeatedly to form the polynomial representations for the elements of GF(2^4). For example,

$$\alpha^5 = \alpha \cdot \alpha^4 = \alpha(1 + \alpha) = \alpha + \alpha^2,$$

$$\alpha^6 = \alpha \cdot \alpha^5 = \alpha(\alpha + \alpha^2) = \alpha^2 + \alpha^3,$$

$$\alpha^7 = \alpha \cdot \alpha^6 = \alpha(\alpha^2 + \alpha^3) = \alpha^3 + \alpha^4 = \alpha^3 + 1 + \alpha = 1 + \alpha + \alpha^3.$$

TABLE 2.8 THREE REPRESENTATIONS FOR THE ELEMENTS OF GF(2^4) GENERATED BY $p(X) = 1 + X + X^4$

Power representation	Polynomial representation	4-Tuple representation
0	0	(0 0 0 0)
1	1	(1 0 0 0)
α	α	(0 1 0 0)
α^2	α^2	(0 0 1 0)
α^3	α^3	(0 0 0 1)
α^4	$1 + \alpha$	(1 1 0 0)
α^5	$\alpha + \alpha^2$	(0 1 1 0)
α^6	$\alpha^2 + \alpha^3$	(0 0 1 1)
α^7	$1 + \alpha + \alpha^3$	(1 1 0 1)
α^8	$1 + \alpha^2$	(1 0 1 0)
α^9	$\alpha + \alpha^3$	(0 1 0 1)
α^{10}	$1 + \alpha + \alpha^2$	(1 1 1 0)
α^{11}	$\alpha + \alpha^2 + \alpha^3$	(0 1 1 1)
α^{12}	$1 + \alpha + \alpha^2 + \alpha^3$	(1 1 1 1)
α^{13}	$1 + \alpha^2 + \alpha^3$	(1 0 1 1)
α^{14}	$1 + \alpha^3$	(1 0 0 1)

To multiply two elements α^i and α^j, we simply add their exponents and use the fact that $\alpha^{15} = 1$. For example, $\alpha^5 \cdot \alpha^7 = \alpha^{12}$ and $\alpha^{12} \cdot \alpha^7 = \alpha^{19} = \alpha^4$. Dividing α^j by α^i, we simply multiply α^j by the multiplicative inverse α^{15-i} of α^i. For example, $\alpha^4/\alpha^{12} = \alpha^4 \cdot \alpha^3 = \alpha^7$ and $\alpha^{12}/\alpha^5 = \alpha^{12} \cdot \alpha^{10} = \alpha^{22} = \alpha^7$. To add α^i and α^j, we use their polynomial representations in Table 2.8. Thus,

$$\alpha^5 + \alpha^7 = (\alpha + \alpha^2) + (1 + \alpha + \alpha^3) = 1 + \alpha^2 + \alpha^3 = \alpha^{13}$$

$$1 + \alpha^5 + \alpha^{10} = 1 + (\alpha + \alpha^2) + (1 + \alpha + \alpha^2) = 0.$$

There is another useful representation for the field elements in GF(2^m). Let $a_0 + a_1\alpha + a_2\alpha^2 + \cdots + a_{m-1}\alpha^{m-1}$ be the polynomial representation of a field element β. Then we can represent β by an ordered sequence of m components, called an *m-tuple*, as follows:

$$(a_0, a_1, a_2, \ldots, a_{m-1}),$$

where the m components are simply the m coefficients of the polynomial representation of β. Clearly, we see that there is one-to-one correspondence between this m-tuple and the polynomial representation of β. The zero element 0 of GF(2^m) is represented by the zero m-tuple $(0, 0, \ldots, 0)$. Let $(b_0, b_1, \ldots, b_{m-1})$ be the m-tuple representation of γ in GF(2^m). Adding β and γ, we simply add the corresponding components of their m-tuple representations as follows:

$$(a_0 + b_0, a_1 + b_1, \ldots, a_{m-1} + b_{m-1}),$$

where $a_i + b_i$ is carried out in modulo-2 addition. Obviously, the components of the resultant m-tuple are the coefficients of the polynomial representation for $\beta + \gamma$. All three representations for the elements of GF(2^4) are given in Table 2.8.

Galois fields of 2^m elements with $m = 2$ to 10 are given in Appendix A.

2.5 BASIC PROPERTIES OF GALOIS FIELD GF(2^m)

In ordinary algebra we often see that a polynomial with real coefficients has roots not from the field of real numbers but from the field of complex numbers that contains the field of real numbers as a subfield. For example, the polynomial $X^2 + 6X + 25$ does not have roots from the field of real numbers but has two complex conjugate roots, $-3 + 4i$ and $-3 - 4i$, where $i = \sqrt{-1}$. This is also true for polynomials with coefficients from GF(2). In this case, a polynomial with coefficients from GF(2) may not have roots from GF(2) but has roots from an extension field of GF(2). For example, $X^4 + X^3 + 1$ is irreducible over GF(2) and therefore it does not have roots from GF(2). However, it has four roots from the field GF(2^4). If we substitute the elements of GF(2^4) given by Table 2.8 into $X^4 + X^3 + 1$, we find that α^7, α^{11}, α^{13}, and α^{14} are the roots of $X^4 + X^3 + 1$. We may verify this as follows:

$$(\alpha^7)^4 + (\alpha^7)^3 + 1 = \alpha^{28} + \alpha^{21} + 1 = (1 + \alpha^2 + \alpha^3) + (\alpha^2 + \alpha^3) + 1 = 0.$$

Indeed, α^7 is a root for $X^4 + X^3 + 1$. Similarly, we may verify that α^{11}, α^{13}, and α^{14} are the other three roots. Since α^7, α^{11}, α^{13}, and α^{14} are all roots of $X^4 + X^3 + 1$, then $(X + \alpha^7)(X + \alpha^{11})(X + \alpha^{13})(X + \alpha^{14})$ must be equal to $X^4 + X^3 + 1$. To see this, we multiply out the product above using Table 2.8:

$$\begin{aligned}(X + \alpha^7)(X + \alpha^{11})(X + \alpha^{13})(X + \alpha^{14}) &= [X^2 + (\alpha^7 + \alpha^{11})X + \alpha^{18}][X^2 + (\alpha^{13} + \alpha^{14})X + \alpha^{27}] \\ &= (X^2 + \alpha^8 X + \alpha^3)(X^2 + \alpha^2 X + \alpha^{12}) \\ &= X^4 + (\alpha^8 + \alpha^2)X^3 + (\alpha^{12} + \alpha^{10} + \alpha^3)X^2 + (\alpha^{20} + \alpha^5)X + \alpha^{15} \\ &= X^4 + X^3 + 1.\end{aligned}$$

Let $f(X)$ be a polynomial with coefficients from GF(2). If β, an element in GF(2^m), is a root of $f(X)$, the polynomial $f(X)$ may have other roots from GF(2^m). Then, what are these roots? This is answered by the following theorem.

Theorem 2.7. Let $f(X)$ be a polynomial with coefficients from GF(2). Let β be an element in an extension field of GF(2). If β is a root of $f(X)$, then for any $l \geq 0$, β^{2^l} is also a root of $f(X)$.

Proof. From (2.10), we have

$$[f(X)]^{2^l} = f(X^{2^l}).$$

Substituting β into the equation above, we obtain

$$[f(\beta)]^{2^l} = f(\beta^{2^l}).$$

Since $f(\beta) = 0$, $f(\beta^{2^l}) = 0$. Therefore, β^{2^l} is also a root of $f(X)$. Q.E.D.

The element β^{2^l} is called a *conjugate* of β. Theorem 2.7 says that if β, an element in GF(2^m), is a root of a polynomial $f(X)$ over GF(2), then all the distinct conjugates of β, also elements in GF(2^m), are roots of $f(X)$. For example, the polynomial $f(X) = 1 + X^3 + X^4 + X^5 + X^6$ has α^4, an element in GF(2^4) given by Table 2.8, as a root.

To verify this, we use Table 2.8 and the fact that $\alpha^{15} = 1$,

$$\begin{aligned}f(\alpha^4) &= 1 + \alpha^{12} + \alpha^{16} + \alpha^{20} + \alpha^{24} = 1 + \alpha^{12} + \alpha + \alpha^5 + \alpha^9 \\ &= 1 + (1 + \alpha + \alpha^2 + \alpha^3) + \alpha + (\alpha + \alpha^2) + (\alpha + \alpha^3) = 0.\end{aligned}$$

The conjugates of α^4 are

$$(\alpha^4)^2 = \alpha^8, \qquad (\alpha^4)^{2^2} = \alpha^{16} = \alpha, \qquad (\alpha^4)^{2^3} = \alpha^{32} = \alpha^2.$$

[Note that $(\alpha^4)^{2^4} = \alpha^{64} = \alpha^4$.] It follows from Theorem 2.7 that α^8, α, and α^2 must be also roots of $f(X) = 1 + X^3 + X^4 + X^5 + X^6$. We can check that α^5 and its conjugate α^{10} are roots of $f(X) = 1 + X^3 + X^4 + X^5 + X^6$.

Let β be a nonzero element in the field GF(2^m). It follows from Theorem 2.4 that

$$\beta^{2^m - 1} = 1.$$

Adding 1 to both sides of $\beta^{2^m-1} = 1$, we obtain

$$\beta^{2^m - 1} + 1 = 0.$$

This says that β is a root of the polynomial $X^{2^m-1} + 1$. Hence, every nonzero element of GF(2^m) is a root of $X^{2^m-1} + 1$. Since the degree of $X^{2^m-1} + 1$ is $2^m - 1$, the $2^m - 1$ nonzero elements of GF(2^m) form all the roots of $X^{2^m-1} + 1$. Summarizing the result above, we obtain Theorem 2.8.

Theorem 2.8. The $2^m - 1$ nonzero elements of GF(2^m) form all the roots of $X^{2^m-1} + 1$.

Since the zero element 0 of GF(2^m) is the root of X, Theorem 2.8 has the following corollary:

Corollary 2.8.1. The elements of GF(2^m) form all the roots of $X^{2^m} + X$.

Since any element β in GF(2^m) is a root of the polynomial $X^{2^m} + X$, β may be a root of a polynomial over GF(2) with a degree less than 2^m. Let $\phi(X)$ be the polynomial of *smallest degree* over GF(2) such that $\phi(\beta) = 0$. [We can easily prove that $\phi(X)$ is unique.] This polynomial $\phi(X)$ is called the *minimal polynomial* of β. For example, the minimal polynomial of the zero element 0 of GF(2^m) is X and the minimal polynomial of the unit element 1 is $X + 1$. Next, a number of properties of minimal polynomials are derived.

Theorem 2.9. The minimal polynomial $\phi(X)$ of a field element β is irreducible.

Proof. Suppose that $\phi(X)$ is not irreducible and that $\phi(X) = \phi_1(X)\phi_2(X)$, where both $\phi_1(X)$ and $\phi_2(X)$ have degrees greater than 0 and less than the degree of $\phi(X)$. Since $\phi(\beta) = \phi_1(\beta)\phi_2(\beta) = 0$, either $\phi_1(\beta) = 0$ or $\phi_2(\beta) = 0$. This contradicts the hypothesis that $\phi(X)$ is a polynomial of smallest degree such that $\phi(\beta) = 0$. Therefore, $\phi(X)$ must be irreducible. Q.E.D.

Theorem 2.10. Let $f(X)$ be a polynomial over GF(2). Let $\phi(X)$ be the minimal polynomial of a field element β. If β is a root of $f(X)$, then $f(X)$ is divisible by $\phi(X)$.

Proof. Dividing $f(X)$ by $\phi(X)$, we obtain

$$f(X) = a(X)\phi(X) + r(X),$$

where the degree of the remainder $r(X)$ is less than the degree of $\phi(X)$. Substituting β into the equation above and using the fact that $f(\beta) = \phi(\beta) = 0$, we have $r(\beta) = 0$. If $r(X) \neq 0$, $r(X)$ would be a polynomial of lower degree than $\phi(X)$, which has β as a root. This is a contradiction to the fact that $\phi(X)$ is the minimal polynomial of β. Hence, $r(X)$ must be identical to 0 and $\phi(X)$ divides $f(X)$. Q.E.D.

It follows from Corollary 2.8.1 and Theorem 2.10 that we have the following result:

Theorem 2.11. The minimal polynomial $\phi(X)$ of an element β in GF(2^m) divides $X^{2^m} + X$.

Theorem 2.11 says that all the roots of $\phi(X)$ are from GF(2^m). Then, what are the roots of $\phi(X)$? This will be answered by the next two theorems.

Theorem 2.12. Let $f(X)$ be an irreducible polynomial over GF(2). Let β be an element in GF(2^m). Let $\phi(X)$ be the minimal polynomial of β. If $f(\beta) = 0$, then $\phi(X) = f(X)$.

Proof. It follows from Theorem 2.10 that $\phi(X)$ divides $f(X)$. Since $\phi(X) \neq 1$ and $f(X)$ is irreducible, we must have $\phi(X) = f(X)$. Q.E.D.

Theorem 2.12 says that if an irreducible polynomial has β as a root, it is the minimal polynomial $\phi(X)$ of β. It follows from Theorem 2.7 that β and its conjugates $\beta^2, \beta^{2^2}, \ldots, \beta^{2^i}, \ldots$ are roots of $\phi(X)$. Let e be the smallest integer such that $\beta^{2^e} = \beta$. Then $\beta^2, \beta^{2^2}, \ldots, \beta^{2^{e-1}}$ are all the distinct conjugates of β (see Problem 2.14). Since $\beta^{2^m} = \beta$, $e \leq m$.

Theorem 2.13. Let β be an element in GF(2^m) and let e be the smallest non-negative integer such that $\beta^{2^e} = \beta$. Then

$$f(X) = \prod_{i=0}^{e-1} (X + \beta^{2^i})$$

is an irreducible polynomial over GF(2).

Proof. Consider

$$[f(X)]^2 = \left[\prod_{i=0}^{e-1} (X + \beta^{2^i})\right]^2 = \prod_{i=0}^{e-1} (X + \beta^{2^i})^2.$$

Since $(X + \beta^{2^i})^2 = X^2 + (\beta^{2^i} + \beta^{2^i})X + \beta^{2^{i+1}} = X^2 + \beta^{2^{i+1}}$,

$$[f(X)]^2 = \prod_{i=0}^{e-1} (X^2 + \beta^{2^{i+1}}) = \prod_{i=1}^{e} (X^2 + \beta^{2^i})$$

$$= \left[\prod_{i=1}^{e-1} (X^2 + \beta^{2^i})\right](X^2 + \beta^{2^e}).$$

Since $\beta^{2^e} = \beta$, then

$$[f(X)]^2 = \prod_{i=0}^{e-1} (X^2 + \beta^{2^i}) = f(X^2). \tag{2.21}$$

Let $f(X) = f_0 + f_1 X + \cdots + f_e X^e$, where $f_e = 1$. Expand

$$[f(X)]^2 = (f_0 + f_1 X + \cdots + f_e X^e)^2$$

$$= \sum_{i=0}^{e} f_i^2 X^{2i} + (1 + 1) \sum_{i=0}^{e} \sum_{\substack{j=0 \\ i \neq j}}^{e} f_i f_j X^{i+j} = \sum_{i=0}^{e} f_i^2 X^{2i}. \tag{2.22}$$

From (2.21) and (2.22), we obtain

$$\sum_{i=0}^{e} f_i X^{2i} = \sum_{i=0}^{e} f_i^2 X^{2i}.$$

Then, for $0 \leq i \leq e$, we must have

$$f_i = f_i^2.$$

This holds only when $f_i = 0$ or 1. Therefore, $f(X)$ has coefficients from GF(2).

Now suppose that $f(X)$ is not irreducible over GF(2) and $f(X) = a(X)b(X)$. Since $f(\beta) = 0$, either $a(\beta) = 0$ or $b(\beta) = 0$. If $a(\beta) = 0$, $a(X)$ has $\beta, \beta^2, \ldots, \beta^{2^{e-1}}$ as roots, so $a(X)$ has degree e and $a(X) = f(X)$. Similarly, if $b(\beta) = 0$, $b(X) = f(X)$. Therefore, $f(X)$ must be irreducible. Q.E.D.

A direct consequence of Theorems 2.12 and 13 is Theorem 2.14.

Theorem 2.14. Let $\phi(X)$ be the minimal polynomial of an element β in GF(2^m). Let e be the smallest integer such that $\beta^{2^e} = \beta$. Then

$$\phi(X) = \prod_{i=0}^{e-1} (X + \beta^{2^i}). \tag{2.23}$$

Example 2.7

Consider the Galois field GF(2^4) given by Table 2.8. Let $\beta = \alpha^3$. The conjugates of β are

$$\beta^2 = \alpha^6, \qquad \beta^{2^2} = \alpha^{12}, \qquad \beta^{2^3} = \alpha^{24} = \alpha^9.$$

The minimal polynomial of $\beta = \alpha^3$ is then

$$\phi(X) = (X + \alpha^3)(X + \alpha^6)(X + \alpha^{12})(X + \alpha^9).$$

Multiplying out the right-hand side of the equation above with the aid of Table 2.8, we obtain

$$\begin{aligned}\phi(X) &= [X^2 + (\alpha^3 + \alpha^6)X + \alpha^9][X^2 + (\alpha^{12} + \alpha^9)X + \alpha^{21}] \\ &= (X^2 + \alpha^2 X + \alpha^9)(X^2 + \alpha^8 X + \alpha^6) \\ &= X^4 + (\alpha^2 + \alpha^8)X^3 + (\alpha^6 + \alpha^{10} + \alpha^9)X^2 + (\alpha^{17} + \alpha^8)X + \alpha^{15} \\ &= X^4 + X^3 + X^2 + X + 1.\end{aligned}$$

There is another way of finding the minimal polynomial of a field element, which is illustrated by the following example.

Example 2.8

Suppose that we want to determine the minimal polynomial $\phi(X)$ of $\gamma = \alpha^7$ in GF(2^4). The distinct conjugates of γ are

$$\gamma^2 = \alpha^{14}, \qquad \gamma^{2^2} = \alpha^{28} = \alpha^{13}, \qquad \gamma^{2^3} = \alpha^{56} = \alpha^{11}.$$

Hence, $\phi(X)$ has degree 4 and must be of the following form:

$$\phi(X) = a_0 + a_1X + a_2X^2 + a_3X^3 + X^4.$$

Substituting γ into $\phi(X)$, we have

$$\phi(\gamma) = a_0 + a_1\gamma + a_2\gamma^2 + a_3\gamma^3 + \gamma^4 = 0.$$

Using the polynomial representations for $\gamma, \gamma^2, \gamma^3$, and γ^4 in the equation above, we obtain the following:

$$a_0 + a_1(1 + \alpha + \alpha^3) + a_2(1 + \alpha^3) + a_3(\alpha^2 + \alpha^3) + (1 + \alpha^2 + \alpha^3) = 0$$
$$(a_0 + a_1 + a_2 + 1) + a_1\alpha + (a_3 + 1)\alpha^2 + (a_1 + a_2 + a_3 + 1)\alpha^3 = 0.$$

For the equality above to be true, we must have the coefficients equal to zero,

$$\begin{aligned} a_0 + a_1 + a_2 \quad &+ 1 = 0, \\ a_1 \quad &= 0, \\ a_3 &+ 1 = 0, \\ a_1 + a_2 + a_3 &+ 1 = 0. \end{aligned}$$

Solving the linear equations above, we obtain $a_0 = 1$, $a_1 = a_2 = 0$, and $a_3 = 1$. Therefore, the minimal polynomial of $\gamma = \alpha^7$ is $\phi(X) = 1 + X^3 + X^4$. All the minimal polynomials of the elements in GF(2^4) are given by Table 2.9.

TABLE 2.9 MINIMAL POLYNOMIALS OF THE ELEMENTS IN GF(2^4) GENERATED BY $p(X) = X^4 + X + 1$

Conjugate roots	Minimal polynomials
0	X
1	$X + 1$
$\alpha, \alpha^2, \alpha^4, \alpha^8$	$X^4 + X + 1$
$\alpha^3, \alpha^6, \alpha^9, \alpha^{12}$	$X^4 + X^3 + X^2 + X + 1$
α^5, α^{10}	$X^2 + X + 1$
$\alpha^7, \alpha^{11}, \alpha^{13}, \alpha^{14}$	$X^4 + X^3 + 1$

A direct consequence of Theorem 2.14 is Theorem 2.15.

Theorem 2.15. Let $\phi(X)$ be the minimal polynomial of an element β in GF(2^m). Let e be the degree of $\phi(X)$. Then e is the smallest integer such that $\beta^{2^e} = \beta$. Moreover, $e \leq m$.

In particular, the degree of the minimal polynomial of any element in GF(2^m) divides m. The proof of this property is omitted here. Table 2.9 shows that the degree of the minimal polynomial of each element in GF(2^4) divides 4. Minimal polynomials of the elements in GF(2^m) for $m = 2$ to 10 are given in Appendix B.

In the construction of the Galois field GF(2^m), we use a primitive polynomial $p(X)$ of degree m and require that the element α be a root of $p(X)$. Since the powers of α generate all the nonzero elements of GF(2^m), α is a primitive element. In fact,

all the conjugates of α are primitive elements of GF(2^m). To see this, let n be the order of α^{2^l} for $l > 0$. Then

$$(\alpha^{2^l})^n = \alpha^{n2^l} = 1.$$

Also, it follows from Theorem 2.5 that n divides $2^m - 1$,

$$2^m - 1 = k \cdot n. \tag{2.24}$$

Since α is a primitive element of GF(2^m), its order is $2^m - 1$. For $\alpha^{n2^l} = 1$, $n2^l$ must be a multiple of $2^m - 1$. Since 2^l and $2^m - 1$ are relatively prime, n must be divisible by $2^m - 1$, say

$$n = q \cdot (2^m - 1). \tag{2.25}$$

From (2.24) and (2.25) we conclude that $n = 2^m - 1$. Consequently, α^{2^l} is also a primitive element of GF(2^m). In general, we have the following theorem:

Theorem 2.16. If β is a primitive element of GF(2^m), all its conjugates β^2, $\beta^{2^2}, \ldots$ are also primitive elements of GF(2^m).

Example 2.9

Consider the field GF(2^4) given by Table 2.8. The powers of $\beta = \alpha^7$ are

$$\beta^0 = 1, \quad \beta^1 = \alpha^7, \quad \beta^2 = \alpha^{14}, \quad \beta^3 = \alpha^{21} = \alpha^6, \quad \beta^4 = \alpha^{28} = \alpha^{13},$$
$$\beta^5 = \alpha^{35} = \alpha^5, \quad \beta^6 = \alpha^{42} = \alpha^{12}, \quad \beta^7 = \alpha^{49} = \alpha^4, \quad \beta^8 = \alpha^{56} = \alpha^{11},$$
$$\beta^9 = \alpha^{63} = \alpha^3, \quad \beta^{10} = \alpha^{70} = \alpha^{10}, \quad \beta^{11} = \alpha^{77} = \alpha^2, \quad \beta^{12} = \alpha^{84} = \alpha^9,$$
$$\beta^{13} = \alpha^{91} = \alpha, \quad \beta^{14} = \alpha^{98} = \alpha^8, \quad \beta^{15} = \alpha^{105} = 1.$$

Clearly, the powers of $\beta = \alpha^7$ generate all the nonzero elements of GF(2^4), so $\beta = \alpha^7$ is a primitive element of GF(2^7). The conjugates of $\beta = \alpha^7$ are

$$\beta^2 = \alpha^{14}, \qquad \beta^{2^2} = \alpha^{13}, \qquad \beta^{2^3} = \alpha^{11}.$$

We may readily check that they are all primitive elements of GF(2^m).

A more general form of Theorem 2.16 is Theorem 2.17.

Theorem 2.17. If β is an element of order n in GF(2^m), all its conjugates have the same order n. (The proof is left as an exercise.)

Example 2.10

Consider the element α^5 in GF(2^4) given by Table 2.8. Since $(\alpha^5)^{2^2} = \alpha^{20} = \alpha^5$, the only conjugate of α^5 is α^{10}. Both α^5 and α^{10} have order $n = 3$. The minimal polynomial of α^5 and α^{10} is $X^2 + X + 1$, whose degree is a factor of $m = 4$. The conjugates of α^3 are α^6, α^9, and α^{12}. They all have order $n = 5$.

2.6 COMPUTATIONS USING GALOIS FIELD GF(2^m) ARITHMETIC

Here we perform some example computations using arithmetic over GF(2^m). Consider the following linear equations over GF(2^4) (see Table 2.8):

$$\begin{aligned} X + \alpha^7 Y &= \alpha^2, \\ \alpha^{12}X + \alpha^8 Y &= \alpha^4. \end{aligned} \tag{2.26}$$

Multiplying the second equation by α^3 gives

$$X + \alpha^7 Y = \alpha^2,$$
$$X + \alpha^{11} Y = \alpha^7.$$

By adding the two equations above, we get

$$(\alpha^7 + \alpha^{11})Y = \alpha^2 + \alpha^7,$$
$$\alpha^8 Y = \alpha^{12},$$
$$Y = \alpha^4.$$

Substituting $Y = \alpha^4$ into the first equation of (2.26), we obtain $X = \alpha^9$. Thus, the solution for the equations of (2.26) is $X = \alpha^9$ and $Y = \alpha^4$.

Alternatively, the equations of (2.26) could be solved by using Cramer's rule:

$$X = \frac{\begin{vmatrix} \alpha^2 & \alpha^7 \\ \alpha^4 & \alpha^8 \end{vmatrix}}{\begin{vmatrix} 1 & \alpha^7 \\ \alpha^{12} & \alpha^8 \end{vmatrix}} = \frac{\alpha^{10} + \alpha^{11}}{\alpha^8 + \alpha^{19}} = \frac{1 + \alpha^3}{\alpha + \alpha^2} = \frac{\alpha^{14}}{\alpha^5} = \alpha^9,$$

$$Y = \frac{\begin{vmatrix} 1 & \alpha^2 \\ \alpha^{12} & \alpha^4 \end{vmatrix}}{\begin{vmatrix} 1 & \alpha^7 \\ \alpha^{12} & \alpha^8 \end{vmatrix}} = \frac{\alpha^4 + \alpha^{14}}{\alpha^8 + \alpha^{19}} = \frac{\alpha + \alpha^3}{\alpha + \alpha^2} = \frac{\alpha^9}{\alpha^5} = \alpha^4.$$

As one more example, suppose that we want to solve the equation

$$f(X) = X^2 + \alpha^7 X + \alpha = 0$$

over GF(2^4). The quadratic formula will not work because it requires dividing by 2, and in this field, $2 = 0$. If $f(X) = 0$ has any solutions in GF(2^4), the solutions can be found simply by substituting all the elements of Table 2.8 for X. By doing so, we would find that $f(\alpha^6) = 0$ and $f(\alpha^{10}) = 0$, since

$$f(\alpha^6) = (\alpha^6)^2 + \alpha^7 \cdot \alpha^6 + \alpha = \alpha^{12} + \alpha^{13} + \alpha = 0,$$
$$f(\alpha^{10}) = (\alpha^{10})^2 + \alpha^7 \cdot \alpha^{10} + \alpha = \alpha^5 + \alpha^2 + \alpha = 0.$$

Thus, α^6 and α^{10} are the roots of $f(X)$ and $f(X) = (X + \alpha^6)(X + \alpha^{10})$.

The computations above are typical of those required for decoding a class of block codes, known as BCH codes, and they can be programmed quite easily on a general-purpose computer. It is also a simple matter to build a computer that can do this kind of arithmetic.

2.7 VECTOR SPACES

Let V be a set of elements on which a binary operation called addition $+$ is defined. Let F be a field. A multiplication operation, denoted by $\cdot$, between the elements in F and elements in V is also defined. The set V is called a *vector space* over the field F if it satisfies the following conditions:

(i) V is a commutative group under addition.

(ii) For any element a in F and any element $\mathbf{v}$ in V, $a \cdot \mathbf{v}$ is an element in V.

(iii) (Distributive Laws) For any elements $\mathbf{u}$ and $\mathbf{v}$ in V and any elements a and b in F,

$$a \cdot (\mathbf{u} + \mathbf{v}) = a \cdot \mathbf{u} + a \cdot \mathbf{v},$$
$$(a + b) \cdot \mathbf{v} = a \cdot \mathbf{v} + b \cdot \mathbf{v}.$$

(iv) (Associative Law) For any $\mathbf{v}$ in V and any a and b in F,

$$(a \cdot b) \cdot \mathbf{v} = a \cdot (b \cdot \mathbf{v}).$$

(v) Let 1 be the unit element of F. Then, for any $\mathbf{v}$ in V, $1 \cdot \mathbf{v} = \mathbf{v}$.

The elements of V are called *vectors* and the elements of the field F are called *scalars*. The addition on V is called a *vector addition* and the multiplication that combines a scalar in F and a vector in V into a vector in V is referred to as *scalar multiplication* (or *product*). The additive identity of V is denoted by $\mathbf{0}$.

Some basic properties of a vector space V over a field F can be derived from the definition above.

Property I. Let 0 be the zero element of the field F. For any vector $\mathbf{v}$ in V, $0 \cdot \mathbf{v} = \mathbf{0}$.

Proof. Since $1 + 0 = 1$ in F, we have $1 \cdot \mathbf{v} = (1 + 0) \cdot \mathbf{v} = 1 \cdot \mathbf{v} + 0 \cdot \mathbf{v}$. Using condition (v) of the definition of a vector space given above, we obtain $\mathbf{v} = \mathbf{v} + 0 \cdot \mathbf{v}$. Let $-\mathbf{v}$ be the additive inverse of $\mathbf{v}$. Adding $-\mathbf{v}$ to both sides of $\mathbf{v} = \mathbf{v} + 0 \cdot \mathbf{v}$, we have

$$\mathbf{0} = \mathbf{0} + 0 \cdot \mathbf{v}$$
$$\mathbf{0} = 0 \cdot \mathbf{v}.$$ Q.E.D.

Property II. For any scalar c in F, $c \cdot \mathbf{0} = \mathbf{0}$. (The proof is left as an exercise.)

Property III. For any scalar c in F and any vector $\mathbf{v}$ in V,

$$(-c) \cdot \mathbf{v} = c \cdot (-\mathbf{v}) = -(c \cdot \mathbf{v})$$

That is, $(-c) \cdot \mathbf{v}$ or $c \cdot (-\mathbf{v})$ is the additive inverse of the vector $c \cdot \mathbf{v}$. (The proof is left as an exercise.)

Next, we present a very useful vector space over GF(2) which plays a central role in coding theory. Consider an ordered sequence of n components,

$$(a_0, a_1, \ldots, a_{n-1}),$$

where each component a_i is an element from the binary field GF(2) (i.e., $a_i = 0$ or 1). This sequence is generally called an *n-tuple* over GF(2). Since there are two choices for each a_i, we can construct 2^n distinct n-tuples. Let V_n denote this set of 2^n distinct n-tuples. Now, we define an addition $+$ on V_n as the following: For any $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ and $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ in V_n,

$$\mathbf{u} + \mathbf{v} = (u_0 + v_0, u_1 + v_1, \ldots, u_{n-1} + v_{n-1}), \tag{2.27}$$

where $u_i + v_i$ is carried out in modulo-2 addition. Clearly, $\mathbf{u} + \mathbf{v}$ is also an n-tuple over GF(2). Hence, V_n is closed under the addition defined by (2.27). We can readily verify that V_n is a commutative group under the addition defined by (2.27). First

we see that the all-zero n-tuple $\mathbf{0} = (0, 0, \ldots, 0)$ is the additive identity. For any $\mathbf{v}$ in V_n,

$$\begin{aligned}\mathbf{v} + \mathbf{v} &= (v_0 + v_0, v_1 + v_1, \ldots, v_{n-1} + v_{n-1}) \\ &= (0, 0, \ldots, 0) = \mathbf{0}.\end{aligned}$$

Hence, the additive inverse of each n-tuple in V_n is itself. Since modulo-2 addition is commutative and associative, we can easily check that the addition defined by (2.27) is also commutative and associative. Therefore, V_n is a commutative group under the addition defined by (2.27).

Next we define scalar multiplication of an n-tuple $\mathbf{v}$ in V_n by an element a from GF(2) as follows:

$$a\cdot(v_0, v_1, \ldots, v_{n-1}) = (a\cdot v_0, a\cdot v_1, \ldots, a\cdot v_{n-1}), \tag{2.28}$$

where $a\cdot v_i$ is carried out in modulo-2 multiplication. Clearly, $a\cdot(v_0, v_1, \ldots, v_{n-1})$ is also an n-tuple in V_n. If $a = 1$,

$$\begin{aligned}1\cdot(v_0, v_1, \ldots, v_{n-1}) &= (1\cdot v_0, 1\cdot v_1, \ldots, 1\cdot v_{n-1}) \\ &= (v_0, v_1, \ldots, v_{n-1}).\end{aligned}$$

We can easily show that the vector addition and scalar multiplication defined by (2.27) and (2.28), respectively, satisfy the distributive and associative laws. Therefore, the set V_n of all n-tuples over GF(2) forms a vector space over GF(2).

Example 2.11

Let $n = 5$. The vector space V_5 of all 5-tuples over GF(2) consists of the following 32 vectors:

$$\begin{array}{llll}
(0\ 0\ 0\ 0\ 0), & (0\ 0\ 0\ 0\ 1), & (0\ 0\ 0\ 1\ 0), & (0\ 0\ 0\ 1\ 1), \\
(0\ 0\ 1\ 0\ 0), & (0\ 0\ 1\ 0\ 1), & (0\ 0\ 1\ 1\ 0), & (0\ 0\ 1\ 1\ 1), \\
(0\ 1\ 0\ 0\ 0), & (0\ 1\ 0\ 0\ 1), & (0\ 1\ 0\ 1\ 0), & (0\ 1\ 0\ 1\ 1), \\
(0\ 1\ 1\ 0\ 0), & (0\ 1\ 1\ 0\ 1), & (0\ 1\ 1\ 1\ 0), & (0\ 1\ 1\ 1\ 1), \\
(1\ 0\ 0\ 0\ 0), & (1\ 0\ 0\ 0\ 1), & (1\ 0\ 0\ 1\ 0), & (1\ 0\ 0\ 1\ 1), \\
(1\ 0\ 1\ 0\ 0), & (1\ 0\ 1\ 0\ 1), & (1\ 0\ 1\ 1\ 0), & (1\ 0\ 1\ 1\ 1), \\
(1\ 1\ 0\ 0\ 0), & (1\ 1\ 0\ 0\ 1), & (1\ 1\ 0\ 1\ 0), & (1\ 1\ 0\ 1\ 1), \\
(1\ 1\ 1\ 0\ 0), & (1\ 1\ 1\ 0\ 1), & (1\ 1\ 1\ 1\ 0), & (1\ 1\ 1\ 1\ 1).
\end{array}$$

The vector sum of (1 0 1 1 1) and (1 1 0 0 1) is

$$(1\ 0\ 1\ 1\ 1) + (1\ 1\ 0\ 0\ 1) = (1 + 1, 0 + 1, 1 + 0, 1 + 0, 1 + 1) = (0\ 1\ 1\ 1\ 0).$$

Using the rule of scalar multiplication defined by (2.28), we obtain

$$0\cdot(1\ 1\ 0\ 1\ 0) = (0\cdot 1, 0\cdot 1, 0\cdot 0, 0\cdot 1, 0\cdot 0) = (0\ 0\ 0\ 0\ 0),$$

$$1\cdot(1\ 1\ 0\ 1\ 0) = (1\cdot 1, 1\cdot 1, 1\cdot 0, 1\cdot 1, 1\cdot 0) = (1\ 1\ 0\ 1\ 0).$$

The vector space of all n-tuples over any field F can be constructed in a similar manner. However, in this book, we are concerned only with the vector space of all n-tuples over GF(2) or over an extension field of GF(2) [e.g., GF(2^m)].

V being a vector space over a field F, it may happen that a subset S of V is also a vector space over F. Such a subset is called a *subspace* of V.

Theorem 2.18. Let S be a nonempty subset of a vector space V over a field F. Then S is a subspace of V if the following conditions are satisfied:

(i) For any two vectors $\mathbf{u}$ and $\mathbf{v}$ in S, $\mathbf{u} + \mathbf{v}$ is also a vector in S.
(ii) For any element a in F and any vector $\mathbf{u}$ in S, $a\cdot\mathbf{u}$ is also in S.

Proof. Conditions (i) and (ii) say simply that S is closed under vector addition and scalar multiplication of V. Condition (ii) ensures that, for any vector $\mathbf{v}$ in S, its additive inverse $(-1)\cdot\mathbf{v}$ is also in S. Then, $\mathbf{v} + (-1)\cdot\mathbf{v} = \mathbf{0}$ is also in S. Therefore, S is a subgroup of V. Since the vectors of S are also vectors of V, the associative and distributive laws must hold for S. Hence, S is a vector space over F and is a subspace of V. Q.E.D.

Example 2.12

Consider the vector space V_5 of all 5-tuples over GF(2) given in Example 2.11. The set

$$\{(0\ 0\ 0\ 0\ 0),\ (0\ 0\ 1\ 1\ 1),\ (1\ 1\ 0\ 1\ 0),\ (1\ 1\ 1\ 0\ 1)\}$$

satisfies both conditions of Theorem 2.18, so it is a subspace of V_5.

Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ be k vectors in a vector space V over a field F. Let $a_1, a_2, \ldots, a_k$ be k scalars from F. The sum

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k$$

is called a *linear combination* of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$. Clearly, the sum of two linear combinations of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$,

$$\begin{aligned}&(a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k) + (b_1\mathbf{v}_1 + b_2\mathbf{v}_2 + \cdots + b_k\mathbf{v}_k) \\ &\qquad = (a_1 + b_1)\mathbf{v}_1 + (a_2 + b_2)\mathbf{v}_2 + \cdots + (a_k + b_k)\mathbf{v}_k,\end{aligned}$$

is also a linear combination of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$, and the product of a scalar c in F and a linear combination of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$,

$$c\cdot(a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k) = (c\cdot a_1)\mathbf{v}_1 + (c\cdot a_2)\mathbf{v}_2 + \cdots + (c\cdot a_k)\mathbf{v}_k,$$

is also a linear combination of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$. It follows from Theorem 2.18 that we have the following result.

Theorem 2.19. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ be k vectors in a vector space V over a field F. The set of all linear combinations of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ forms a subspace of V.

Example 2.13

Consider the vector space V_5 of all 5-tuples over GF(2) given by Example 2.11. The linear combinations of (0 0 1 1 1) and (1 1 1 0 1) are

$$0\cdot(0\ 0\ 1\ 1\ 1) + 0\cdot(1\ 1\ 1\ 0\ 1) = (0\ 0\ 0\ 0\ 0),$$

$$0\cdot(0\ 0\ 1\ 1\ 1) + 1\cdot(1\ 1\ 1\ 0\ 1) = (1\ 1\ 1\ 0\ 1),$$

$$1\cdot(0\ 0\ 1\ 1\ 1) + 0\cdot(1\ 1\ 1\ 0\ 1) = (0\ 0\ 1\ 1\ 1),$$

$$1\cdot(0\ 0\ 1\ 1\ 1) + 1\cdot(1\ 1\ 1\ 0\ 1) = (1\ 1\ 0\ 1\ 0).$$

These four vectors form the same subspace given by Example 2.12.

A set of vectors $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ in a vector space V over a field F is said to be *linearly dependent* if and only if there exist k scalars $a_1, a_2, \ldots, a_k$ from F, *not all zero*, such that

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k = \mathbf{0}.$$

A set of vectors, $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$, is said to be *linearly independent* if it is not linearly dependent. That is, if $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ are linearly independent, then

$$a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \cdots + a_k\mathbf{v}_k \neq \mathbf{0}$$

unless $a_1 = a_2 = \cdots = a_k = 0$.

Example 2.14

The vectors (1 0 1 1 0), (0 1 0 0 1), and (1 1 1 1 1) are linearly dependent since

$$1\cdot(1\ 0\ 1\ 1\ 0) + 1\cdot(0\ 1\ 0\ 0\ 1) + 1\cdot(1\ 1\ 1\ 1\ 1) = (0\ 0\ 0\ 0\ 0).$$

However, (1 0 1 1 0), (0 1 0 0 1), and (1 1 0 1 1) are linearly independent. All eight linear combinations of these three vectors are given below:

$$0\cdot(1\ 0\ 1\ 1\ 0) + 0\cdot(0\ 1\ 0\ 0\ 1) + 0\cdot(1\ 1\ 0\ 1\ 1) = (0\ 0\ 0\ 0\ 0),$$
$$0\cdot(1\ 0\ 1\ 1\ 0) + 0\cdot(0\ 1\ 0\ 0\ 1) + 1\cdot(1\ 1\ 0\ 1\ 1) = (1\ 1\ 0\ 1\ 1),$$
$$0\cdot(1\ 0\ 1\ 1\ 0) + 1\cdot(0\ 1\ 0\ 0\ 1) + 0\cdot(1\ 1\ 0\ 1\ 1) = (0\ 1\ 0\ 0\ 1),$$
$$0\cdot(1\ 0\ 1\ 1\ 0) + 1\cdot(0\ 1\ 0\ 0\ 1) + 1\cdot(1\ 1\ 0\ 1\ 1) = (1\ 0\ 0\ 1\ 0),$$
$$1\cdot(1\ 0\ 1\ 1\ 0) + 0\cdot(0\ 1\ 0\ 0\ 1) + 0\cdot(1\ 1\ 0\ 1\ 1) = (1\ 0\ 1\ 1\ 0),$$
$$1\cdot(1\ 0\ 1\ 1\ 0) + 0\cdot(0\ 1\ 0\ 0\ 1) + 1\cdot(1\ 1\ 0\ 1\ 1) = (0\ 1\ 1\ 0\ 1),$$
$$1\cdot(1\ 0\ 1\ 1\ 0) + 1\cdot(0\ 1\ 0\ 0\ 1) + 0\cdot(1\ 1\ 0\ 1\ 1) = (1\ 1\ 1\ 1\ 1),$$
$$1\cdot(1\ 0\ 1\ 1\ 0) + 1\cdot(0\ 1\ 0\ 0\ 1) + 1\cdot(1\ 1\ 0\ 1\ 1) = (0\ 0\ 1\ 0\ 0).$$

A set of vectors is said to *span* a vector space V if every vector in V is a linear combination of the vectors in the set. In any vector space or subspace there exists at least one set B of linearly independent vectors which span the space. This set is called a *basis* (or *base*) of the vector space. The number of vectors in a basis of a vector space is called the *dimension* of the vector space. (Note that the number of vectors in any two bases are the same.)

Consider the vector space V_n of all n-tuples over GF(2). Let us form the following n n-tuples:

$$\mathbf{e}_0 = (1, 0, 0, 0, \ldots, 0, 0)$$
$$\mathbf{e}_1 = (0, 1, 0, 0, \ldots, 0, 0)$$
$$\vdots$$
$$\mathbf{e}_{n-1} = (0, 0, 0, 0, \ldots, 0, 1),$$

where the n-tuple $\mathbf{e}_i$ has only one nonzero component at ith position. Then every n-tuple $(a_0, a_1, a_2, \ldots, a_{n-1})$ in V_n can be expressed as a linear combination of $\mathbf{e}_0$, $\mathbf{e}_1, \ldots, \mathbf{e}_{n-1}$ as follows:

$$(a_0, a_1, a_2, \ldots, a_{n-1}) = a_0\mathbf{e}_0 + a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + \cdots + a_{n-1}\mathbf{e}_{n-1}.$$

Therefore, $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{n-1}$ span the vector space V_n of all n-tuples over GF(2). From the equation above, we also see that $\mathbf{e}_0, \mathbf{e}_1, \ldots, \mathbf{e}_{n-1}$ are linearly independent. Hence, they form a basis for V_n and the dimension of V_n is n. If $k < n$ and $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ are k linearly independent vectors in V_n, then all the linear combinations of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$ of the form

$$\mathbf{u} = c_1\mathbf{v}_1 + c_2\mathbf{v}_2 + \cdots + c_k\mathbf{v}_k$$

form a k-dimensional subspace S of V_n. Since each c_i has two possible values, 0 or 1, there are 2^k possible distinct linear combinations of $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_k$. Thus, S consists of 2^k vectors and is a k-dimensional subspace of V_n.

Let $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ and $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be two n-tuples in V_n. We define the *inner product* (or *dot product*) or $\mathbf{u}$ and $\mathbf{v}$ as

$$\mathbf{u}\cdot\mathbf{v} = u_0\cdot v_0 + u_1\cdot v_1 + \cdots + u_{n-1}\cdot v_{n-1}, \tag{2.29}$$

where $u_i\cdot v_i$ and $u_i\cdot v_i + u_{i+1}\cdot v_{i+1}$ are carried out in modulo-2 multiplication and addition. Hence, the inner product $\mathbf{u}\cdot\mathbf{v}$ is a scalar in GF(2). If $\mathbf{u}\cdot\mathbf{v} = 0$, $\mathbf{u}$ and $\mathbf{v}$ are said to be *orthogonal* to each other. The inner product has the following properties:

(i) $\mathbf{u}\cdot\mathbf{v} = \mathbf{v}\cdot\mathbf{u}$.
(ii) $\mathbf{u}\cdot(\mathbf{v} + \mathbf{w}) = \mathbf{u}\cdot\mathbf{v} + \mathbf{u}\cdot\mathbf{w}$.
(iii) $(a\mathbf{u})\cdot\mathbf{v} = a(\mathbf{u}\cdot\mathbf{v})$.

(The concept of inner product can be generalized to any Galois field.)

Let S be a k-dimensional subspace of V_n and let S_d be the set of vectors in V_n such that, for any $\mathbf{u}$ in S and $\mathbf{v}$ in S_d, $\mathbf{u}\cdot\mathbf{v} = 0$. The set S_d contains at least the all-zero n-tuple $\mathbf{0} = (0, 0, \ldots, 0)$, since for any $\mathbf{u}$ in S, $\mathbf{0}\cdot\mathbf{u} = 0$. Thus, S_d is nonempty. For any element a in GF(2) and any $\mathbf{v}$ in S_d,

$$a\cdot\mathbf{v} = \begin{cases} \mathbf{0} & \text{if } a = 0 \\ \mathbf{v} & \text{if } a = 1. \end{cases}$$

Therefore, $a\cdot\mathbf{v}$ is also in S_d. Let $\mathbf{v}$ and $\mathbf{w}$ be any two vectors in S_d. For any vector $\mathbf{u}$ in S, $\mathbf{u}\cdot(\mathbf{v} + \mathbf{w}) = \mathbf{u}\cdot\mathbf{v} + \mathbf{u}\cdot\mathbf{w} = 0 + 0 = 0$. This says that if $\mathbf{v}$ and $\mathbf{w}$ are orthogonal to $\mathbf{u}$, the vector sum $\mathbf{v} + \mathbf{w}$ is also orthogonal to $\mathbf{u}$. Consequently, $\mathbf{v} + \mathbf{w}$ is a vector in S_d. It follows from Theorem 2.18 that S_d is also a subspace of V_n. This subspace S_d is called the *null* (or *dual*) *space* of S. Conversely, S is also the null space of S_d. The dimension of S_d is given by Theorem 2.20, whose proof is omitted here [2].

Theorem 2.20. Let S be a k-dimensional space of the vector space V_n of all n-tuples over GF(2). The dimension of its null space S_d is $n - k$. In other words, $\dim(S) + \dim(S_d) = n$.

Example 2.15

Consider the vector space V_5 of all 5-tuples over GF(2) given by Example 2.11. The following eight vectors form a three-dimensional subspace S of V_5:

(0 0 0 0 0), (1 1 1 0 0), (0 1 0 1 0), (1 0 0 0 1).
(1 0 1 1 0), (0 1 1 0 1), (1 1 0 1 1), (0 0 1 1 1).

The null space S_d of S consists of the following 4-vectors:

$$(0\ 0\ 0\ 0\ 0),\quad (1\ 0\ 1\ 0\ 1),\quad (0\ 1\ 1\ 1\ 0),\quad (1\ 1\ 0\ 1\ 1).$$

S_d is spanned by (1 0 1 0 1) and (0 1 1 1 0), which are linearly independent. Thus, the dimension of S_d is 2.

All the results presented in this section can be generalized in a straightforward manner to the vector space of all n-tuples over GF(q), where q is a power of prime.

2.8 MATRICES

A $k \times n$ matrix over GF(2) (or over any other field) is a rectangular array with k rows and n columns,

$$\mathbf{G} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,n-1} \\ \cdot & & & & \\ \cdot & & & & \\ \cdot & & & & \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix}, \tag{2.30}$$

where each entry g_{ij} with $0 \leq i < k$ and $0 \leq j < n$ is an element from the binary field GF(2). Observe that the first index i indicates the row containing g_{ij} and the second index j tells which column g_{ij} is in. We shall sometimes abbreviate the matrix of (2.30) by the notation $[g_{ij}]$. We also observe that each row of $\mathbf{G}$ is an n-tuple over GF(2) and each column is a k-tuple over GF(2). The matrix $\mathbf{G}$ can also be represented by its k rows $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix}.$$

If the k ($k \leq n$) rows of $\mathbf{G}$ are linearly independent, then the 2^k linear combinations of these rows form a k-dimensional subspace of the vector space V_n of all the n-tuples over GF(2). This subspace is called the *row space* of $\mathbf{G}$. We may interchange any two rows of $\mathbf{G}$ or add one row to another. These are called *elementary row operations*. Performing elementary row operations on $\mathbf{G}$, we obtain another matrix $\mathbf{G}'$ over GF(2). However, both $\mathbf{G}$ and $\mathbf{G}'$ gave the same row space.

Example 2.16

Consider a 3×6 matrix $\mathbf{G}$ over GF(2),

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

Adding the third row to the first row and interchanging the second and third rows, we obtain the following matrix:

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

Both $\mathbf{G}$ and $\mathbf{G}'$ give the following row space:

$$(0\ 0\ 0\ 0\ 0\ 0),\quad (1\ 0\ 0\ 1\ 0\ 1),\quad (0\ 1\ 0\ 0\ 1\ 1),\quad (0\ 0\ 1\ 1\ 1\ 0),$$
$$(1\ 1\ 0\ 1\ 1\ 0),\quad (1\ 0\ 1\ 0\ 1\ 1),\quad (0\ 1\ 1\ 1\ 0\ 1),\quad (1\ 1\ 1\ 0\ 0\ 0).$$

This is a three-dimensional subspace of the vector space V_6 of all the 6-tuples over GF(2).

Let S be the row space of a $k \times n$ matrix $\mathbf{G}$ over GF(2) whose k rows $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ are linearly independent. Let S_d be the null space of S. Then the dimension of S_d is $n - k$. Let $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{n-k-1}$ be $n - k$ linearly independent vectors in S_d. Clearly, these vectors span S_d. We may form an $(n - k) \times n$ matrix $\mathbf{H}$ using $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{n-k-1}$ as rows:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{h}_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,n-1} \\ h_{10} & h_{11} & \cdots & h_{1,n-1} \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ \cdot & \cdot & & \cdot \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix}.$$

The row space of $\mathbf{H}$ is S_d. Since each row $\mathbf{g}_i$ of $\mathbf{G}$ is a vector in S and each row $\mathbf{h}_j$ of $\mathbf{H}$ is a vector in S_d, the inner product of $\mathbf{g}_i$ and $\mathbf{h}_j$ must be zero (i.e., $\mathbf{g}_i \cdot \mathbf{h}_j = 0$). Since the row space S of $\mathbf{G}$ is the null space of the row space S_d of $\mathbf{H}$, we call S the null (or dual) space of $\mathbf{H}$. Summarizing the results above, we have:

Theorem 2.21. For any $k \times n$ matrix $\mathbf{G}$ over GF(2) with k linearly independent rows, there exists an $(n - k) \times n$ matrix $\mathbf{H}$ over GF(2) with $n - k$ linearly independent rows such that for any row $\mathbf{g}_i$ in $\mathbf{G}$ and any $\mathbf{h}_j$ in $\mathbf{H}$, $\mathbf{g}_i \cdot \mathbf{h}_j = 0$. The row space of $\mathbf{G}$ is the null space of $\mathbf{H}$, and vice versa.

Example 2.17

Consider the following 3×6 matrix over GF(2):

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

The row space of this matrix is the null space of

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

We can easily check that each row of $\mathbf{G}$ is orthogonal to each row of $\mathbf{H}$.

Two matrices can be added if they have the same number of rows and the same number of columns. Adding two $k \times n$ matrices $\mathbf{A} = [a_{ij}]$ and $\mathbf{B} = [b_{ij}]$, we simply add their corresponding entries a_{ij} and b_{ij} as follows:

$$[a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}].$$

Hence, the resultant matrix is also a $k \times n$ matrix. Two matrices can be multiplied provided that the number of columns in the first matrix is equal to the number of rows in the second matrix. Multiplying a $k \times n$ matrix $\mathbf{A} = [a_{ij}]$ by an $n \times l$ matrix $\mathbf{B} = [b_{ij}]$, the product

$$\mathbf{C} = \mathbf{A} \times \mathbf{B} = [c_{ij}]$$

is a $k \times l$ matrix where the entry c_{ij} is equal to the inner product of the ith row $\mathbf{a}_i$ in $\mathbf{A}$ and the jth column $\mathbf{b}_j$ in $\mathbf{B}$, that is,

$$c_{ij} = \mathbf{a}_i \cdot \mathbf{b}_j = \sum_{t=0}^{n-1} a_{it} b_{tj}.$$

Let $\mathbf{G}$ be a $k \times n$ matrix over GF(2). The *transpose* of $\mathbf{G}$, denoted by $\mathbf{G}^T$, is an $n \times k$ matrix whose rows are columns of $\mathbf{G}$ and whose columns are rows of $\mathbf{G}$. A $k \times k$ matrix is called an *identity* matrix if it has 1's on the main diagonal and 0's elsewhere. This matrix is usually denoted by $\mathbf{I}_k$. A *submatrix* of a matrix $\mathbf{G}$ is a matrix that is obtained by striking out given rows or columns of $\mathbf{G}$.

It is straightforward to generalize the concepts and results presented in this section to matrices with entries from GF(q) with q as a powerof prime.

PROBLEMS

2.1. Construct the group under modulo-6 addition.

2.2. Construct the group under modulo-3 multiplication.

2.3. Let m be a positive integer. If m is not a prime, prove that the set $\{1, 2, \ldots, m-1\}$ is not a group under modulo-m multiplication.

2.4. Construct the prime field GF(11) with modulo-11 addition and multiplication. Find all the primitive elements and determine the orders of other elements.

2.5. Let m be a positive integer. If m is not prime, prove that the set $\{0, 1, 2, \ldots, m-1\}$ is not a field under modulo-m addition and multiplication.

2.6. Let λ be the characteristic of a Galois field GF(q). Let 1 be the unit element of GF(q). Show that the sums

$$1, \sum_{i=1}^{2} 1, \sum_{i=1}^{3} 1, \ldots, \sum_{i=1}^{\lambda-1} 1, \sum_{i=1}^{\lambda} 1 = 0$$

form a subfield of GF(q).

2.7. Prove that every finite field has a primitive element.

2.8. Solve the following simultaneous equations of X, Y, Z, and W with modulo-2 arithmetic:

$$\begin{aligned} X + Y \quad\quad + W &= 1, \\ X \quad\quad + Z + W &= 0, \\ X + Y + Z + W &= 1, \\ Y + Z + W &= 0. \end{aligned}$$

2.9. Show that $X^5 + X^3 + 1$ is irreducible over GF(2).

2.10. Let $f(X)$ be a polynomial of degree n over GF(2). The reciprocal of $f(X)$ is defined as

$$f^*(X) = X^n f\left(\frac{1}{X}\right).$$

(a) Prove that $f^*(X)$ is irreducible over GF(2) if and only if $f(X)$ is irreducible over GF(2).

(b) Prove that $f^*(X)$ is primitive if and only if $f(X)$ is primitive.

2.11. Find all the irreducible polynomials of degree 5 over GF(2).

2.12. Construct a table for GF(2^3) based on the primitive polynomial $p(X) = 1 + X + X^3$. Display the power, polynomial, and vector representations of each element. Determine the order of each element.

2.13. Construct a table for GF(2^5) based on the primitive polynomial $p(X) = 1 + X^2 + X^5$. Let α be a primitive element of GF(2^5). Find the minimal polynomials of α^3 and α^7.

2.14. Let β be an element in GF(2^m). Let e be the smallest nonnegative integer such that $\beta^{2^e} = \beta$. Prove that $\beta^2, \beta^{2^2}, \ldots, \beta^{2^{e-1}}$ are all the distinct conjugates of β.

2.15. Prove Theorem 2.17.

2.16. Let α be a primitive element in GF(2^4). Use Table 2.8 to find the roots of $f(X) = X^3 + \alpha^6 X^2 + \alpha^9 X + \alpha^9$.

2.17. Let α be a primitive element in GF(2^4). Use Table 2.8 to solve the following simultaneous equations for X, Y, and Z:

$$\begin{aligned} X + \alpha^5 Y + \quad Z &= \alpha^7, \\ X + \alpha Y + \alpha^7 Z &= \alpha^9, \\ \alpha^2 X + \quad Y + \alpha^6 Z &= \alpha. \end{aligned}$$

2.18. Let V be a vector space over a field F. For any element c in F, prove that $c \cdot \mathbf{0} = \mathbf{0}$.

2.19. Let V be a vector space over a field F. Prove that, for any c in F and any $\mathbf{v}$ in V, $(-c) \cdot \mathbf{v} = c \cdot (-\mathbf{v}) = -(c \cdot \mathbf{v})$.

2.20. Let S be a subset of the vector space V_n of all n-tuples over GF(2). Prove that S is a subspace if for any $\mathbf{u}$ and $\mathbf{v}$ in S, $\mathbf{u} + \mathbf{v}$ is in S.

2.21. Prove that GF(2^m) is a vector space over GF(2).

2.22. Construct the vector space V_5 of all 5-tuples over GF(2). Find a three-dimensional subspace and determine its null space.

2.23. Given the matrices

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

show that the row space of $\mathbf{G}$ is the null space of $\mathbf{H}$, and vice versa.

2.24. Let S_1 and S_2 be two subspaces of a vector V. Show that the intersection of S_1 and S_2 is also a subspace in V.

2.25. Construct the vector space of all 3-tuples over GF(3). Form a two-dimensional subspace and its dual space.

REFERENCES

1. A. A. Albert, *Modern Higher Algebra*, The University of Chicago Press, Chicago, 1937.
2. G. Birkhoff and S. MacLane, *A Survey of Modern Algebra*, Macmillan, New York, 1953.
3. R. D. Carmichael, *Introduction to the Theory of Group of Finite Order*, Ginn & Company, Boston, 1937.
4. J. B. Fraleigh, *A First Course in Abstract Algebra*, 2nd ed., Addison-Wesley, Reading, Mass., 1976.
5. N. Jacobson, *Lectures in Abstract Algebra*, Van Nostrand, Princeton, N.J., Vol. 1, 1951; Vol. 2, 1953; Vol. 3, 1964.
6. R. W. Marsh, *Table of Irreducible Polynomials over GF(2) through Degree 19*, NSA, Washington, D.C., 1957.
7. W. W. Peterson, *Error-Correcting Codes*, MIT Press, Cambridge, Mass., 1961.
8. B. L. Van der Waerden, *Modern Algebra*, Vols. 1 and 2, Ungar, New York, 1949.

3

Linear Block Codes

In this chapter basic concepts of block codes are introduced. For ease of code synthesis and implementation, we restrict our attention to a subclass of the class of all block codes, the *linear block codes*. Since in most present digital computers and digital data communication systems, information is coded in binary digits "0" or "1," we discuss only the linear block codes with symbols from the binary field GF(2). The theory developed for the binary codes can be generalized to codes with symbols from a nonbinary field in a straightforward manner.

First, linear block codes are defined and described in terms of *generator* and *parity-check* matrices. The parity-check equations for a *systematic* code are derived. Encoding of linear block codes is discussed. In Section 3.2 the concept of *syndrome* is introduced. The use of syndrome for error detection and correction is discussed. In Sections 3.3 and 3.4 we define the *minimum distance* of a block code and show that the random-error-detecting and random-error-correcting capabilities of a code are determined by its minimum distance. Probabilities of a decoding error are discussed. In Section 3.5 the *standard array* and its application to the decoding of linear block codes are presented. A general decoder based on the *syndrome decoding* scheme is given. Finally, we conclude the chapter by presenting a class of single-error-correcting linear codes.

References 1 to 4 contain excellent treatments of linear block codes.

3.1 INTRODUCTION TO LINEAR BLOCK CODES

We assume that the output of an information source is a sequence of binary digits "0" or "1." In block coding, this binary information sequence is segmented into *message* blocks of fixed length; each message block, denoted by **u**, consists of k

information digits. There are a total of 2^k distinct messages. The encoder, *according to certain rules*, transforms each input message $\mathbf{u}$ into a binary n-tuple $\mathbf{v}$ with $n > k$. This binary n-tuple $\mathbf{v}$ is referred to as the *code word* (or *code vector*) of the message $\mathbf{u}$. Therefore, corresponding to the 2^k possible messages, there are 2^k code words. This set of 2^k code words is called a *block* code. For a block code to be useful, the 2^k code words must be distinct. Therefore, there should be a one-to-one correspondence between a message $\mathbf{u}$ and its code word $\mathbf{v}$.

For a block code with 2^k code words and length n, unless it has a certain special structure, the encoding apparatus would be prohibitively complex for large k and n since it has to store the 2^k code words of length n in a dictionary. Therefore, we must restrict our attention to block codes that can be mechanized in a practical manner. A desirable structure for a block code to possess is the *linearity*. With this structure in a block code, the encoding complexity will be greatly reduced, as we will see.

Definition 3.1. A block code of length n and 2^k code words is called a *linear* (n, k) code if and only if its 2^k code words form a k-dimensional subspace of the vector space of all the n-tuples over the field GF(2).

In fact, a binary block code is linear if and only if the modulo-2 sum of two code words is also a code word. The block code given in Table 3.1 is a (7, 4) linear code. One can easily check that the sum of any two code words in this code is also a code word.

Since an (n, k) linear code C is a k-dimensional subspace of the vector space V_n of all the binary n-tuples, it is possible to find k linearly independent code words,

TABLE 3.1 LINEAR BLOCK CODE WITH $k = 4$ AND $n = 7$

Messages	Code words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

$\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ in C such that every code word $\mathbf{v}$ in C is a linear combination of these k code words, that is,

$$\mathbf{v} = u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \cdots + u_{k-1}\mathbf{g}_{k-1}, \tag{3.1}$$

where $u_i = 0$ or 1 for $0 \leq i < k$. Let us arrange these k linearly independent code words as the rows of a $k \times n$ matrix as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdots & g_{1,n-1} \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdots & g_{k-1,n-1} \end{bmatrix}, \tag{3.2}$$

where $\mathbf{g}_i = (g_{i0}, g_{i1}, \ldots, g_{i,n-1})$ for $0 \leq i < k$. If $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ is the message to be encoded, the corresponding code word can be given as follows:

$$\begin{aligned} \mathbf{v} &= \mathbf{u} \cdot \mathbf{G} \\ &= (u_0, u_1, \ldots, u_{k-1}) \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} \\ &= u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \cdots + u_{k-1}\mathbf{g}_{k-1}. \end{aligned} \tag{3.3}$$

Clearly, the rows of $\mathbf{G}$ *generate* (or *span*) the (n, k) linear code C. For this reason, the matrix $\mathbf{G}$ is called a *generator* matrix for C. Note that any k linearly independent code words of an (n, k) linear code can be used to form a generator matrix for the code. It follows from (3.3) that an (n, k) linear code is completely specified by the k rows of a generator matrix $\mathbf{G}$. Therefore, the encoder has only to store the k rows of $\mathbf{G}$ and to form a linear combination of these k rows based on the input message $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$.

Example 3.1

The (7, 4) linear code given in Table 3.1 has the following matrix as a generator matrix:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

If $\mathbf{u} = (1\ 1\ 0\ 1)$ is the message to be encoded, its corresponding code word, according to (3.3), would be

$$\begin{aligned} \mathbf{v} &= 1 \cdot \mathbf{g}_0 + 1 \cdot \mathbf{g}_1 + 0 \cdot \mathbf{g}_2 + 1 \cdot \mathbf{g}_3 \\ &= (1\ 1\ 0\ 1\ 0\ 0\ 0) + (0\ 1\ 1\ 0\ 1\ 0\ 0) + (1\ 0\ 1\ 0\ 0\ 0\ 1) \\ &= (0\ 0\ 0\ 1\ 1\ 0\ 1). \end{aligned}$$

A desirable property for a linear block code to possess is the *systematic structure* of the code words as shown in Figure 3.1, where a code word is divided into two parts, the message part and the redundant checking part. The message part consists of k unaltered information (or message) digits and the redundant checking part consists of $n - k$ *parity-check* digits, which are *linear sums* of the information digits. A linear block code with this structure is referred to as a *linear systematic block code*. The (7, 4) code given in Table 3.1 is a linear systematic block code, the rightmost four digits of each code word are identical to the corresponding information digits.

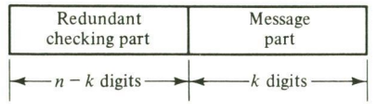

Figure 3.1 Systematic format of a code word.

A linear systematic (n, k) code is completely specified by a $k \times n$ matrix $\mathbf{G}$ of the following form:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ & & & & & & & & \\ & & & & & & & & \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{3.4}$$

$\underbrace{\qquad\qquad\qquad\qquad}_{\mathbf{P}\text{ matrix}}$ $\underbrace{\qquad\qquad\qquad\qquad}_{k \times k\text{ identity matrix}}$

where $p_{ij} = 0$ or 1. Let $\mathbf{I}_k$ denote the $k \times k$ identity matrix. Then $\mathbf{G} = [\mathbf{P}\ \mathbf{I}_k]$. Let $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ be the message to be encoded. The corresponding code word is

$$\begin{aligned} \mathbf{v} &= (v_0, v_1, v_2, \ldots, v_{n-1}) \\ &= (u_0, u_1, \ldots, u_{k-1}) \cdot \mathbf{G}. \end{aligned} \tag{3.5}$$

It follows from (3.4) and (3.5) that the components of $\mathbf{v}$ are

$$v_{n-k+i} = u_i \qquad \text{for } 0 \leq i < k \tag{3.6a}$$

and

$$v_j = u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k-1} p_{k-1,j} \tag{3.6b}$$

for $0 \leq j < n - k$. Equation (3.6a) shows that the rightmost k digits of a code word $\mathbf{v}$ are identical to the information digits $u_0, u_1, \ldots, u_{k-1}$ to be encoded, and (3.6b) shows that the leftmost $n - k$ redundant digits are linear sums of the information digits. The $n - k$ equations given by (3.6b) are called *parity-check equations* of the code.

Example 3.2

The matrix $\mathbf{G}$ given in Example 3.1 is in systematic form. Let $\mathbf{u} = (u_0, u_1, u_2, u_3)$ be the message to be encoded and let $\mathbf{v} = (v_0, v_1, v_2, v_3, v_4, v_5, v_6)$ be the corresponding code word. Then

$$\mathbf{v} = (u_0, u_1, u_2, u_3) \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

By matrix multiplication, we obtain the following digits of the code word $\mathbf{v}$:

$$\begin{aligned} v_6 &= u_3 \\ v_5 &= u_2 \\ v_4 &= u_1 \\ v_3 &= u_0 \\ v_2 &= u_1 + u_2 + u_3 \\ v_1 &= u_0 + u_1 + u_2 \\ v_0 &= u_0 + u_2 + u_3. \end{aligned}$$

The code word corresponding to the message (1 0 1 1) is (1 0 0 1 0 1 1).

There is another useful matrix associated with every linear block code. As stated in Chapter 2, for any $k \times n$ matrix $\mathbf{G}$ with k linearly independent rows, there exists an $(n - k) \times n$ matrix $\mathbf{H}$ with $n - k$ linearly independent rows such that any vector in the row space of $\mathbf{G}$ is orthogonal to the rows of $\mathbf{H}$ and any vector that is orthogonal to the rows of $\mathbf{H}$ is in the row space of $\mathbf{G}$. Hence, we can describe the (n, k) linear code generated by $\mathbf{G}$ in an alternate way as follows: *An n-tuple $\mathbf{v}$ is a code word in the code generated by $\mathbf{G}$ if and only if $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$*. This matrix $\mathbf{H}$ is called a *parity-check matrix* of the code. The 2^{n-k} linear combinations of the rows of matrix $\mathbf{H}$ form an $(n, n - k)$ linear code C_d. This code is the null space of the (n, k) linear code C generated by matrix $\mathbf{G}$ (i.e., for any $\mathbf{v} \in C$ and any $\mathbf{w} \in C_d$, $\mathbf{v} \cdot \mathbf{w} = 0$). C_d is called the *dual* code of C. Therefore, a parity-check matrix for a linear code C is a generator matrix for its dual code C_d.

If the generator matrix of an (n, k) linear code is in the systematic form of (3.4), the parity-check matrix may take the following form:

$$\begin{aligned} \mathbf{H} &= [\mathbf{I}_{n-k}\ \ \mathbf{P}^T] \\ &= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & p_{02} & p_{12} & \cdots & p_{k-1,2} \\ & & & & & & & & \\ & & & & & & & & \\ & & & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}, \end{aligned} \tag{3.7}$$

where $\mathbf{P}^T$ is the transpose of the matrix $\mathbf{P}$. Let $\mathbf{h}_j$ be the jth row of $\mathbf{H}$. We can check readily that the inner product of the ith row of $\mathbf{G}$ given by (3.4) and the jth row of $\mathbf{H}$ given by (3.7) is

$$\mathbf{g}_i \cdot \mathbf{h}_j = p_{ij} + p_{ij} = 0$$

for $0 \leq i < k$ and $0 \leq j < n - k$. This implies that $\mathbf{G} \cdot \mathbf{H}^T = 0$. Also, the $n - k$ rows of $\mathbf{H}$ are linearly independent. Therefore, the $\mathbf{H}$ matrix of (3.7) is a parity-check matrix of the (n, k) linear code generated by the matrix $\mathbf{G}$ of (3.4).

The parity-check equations given by (3.6b) can also be obtained from the parity-check matrix $\mathbf{H}$ of (3.7). Let $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ be the message to be encoded. In systematic form the corresponding code word would be

$$\mathbf{v} = (v_0, v_1, \ldots, v_{n-k-1}, u_0, u_1, \ldots, u_{k-1}).$$

Using the fact that $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$, we obtain

$$v_j + u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k-1} p_{k-1,j} = 0 \tag{3.8}$$

for $0 \leq j < n - k$. Rearranging the equations of (3.8), we obtain the same parity-check equations of (3.6b). Therefore, an (n, k) linear code is completely specified by its parity-check matrix.

Example 3.3

Consider the generator matrix of a (7, 4) linear code given in Example 3.1. The corresponding parity-check matrix is

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

At this point, let us summarize the foregoing results: For any (n, k) linear block code C, there exists a $k \times n$ matrix $\mathbf{G}$ whose row space gives C. Furthermore, there exists an $(n - k) \times n$ matrix $\mathbf{H}$ such that an n-tuple $\mathbf{v}$ is a code word in C if and only if $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$. If $\mathbf{G}$ is of the form given by (3.4), then $\mathbf{H}$ may take the form given by (3.7), and vice versa.

Based on the equations of (3.6a) and (3.6b), the encoding circuit for an (n, k) linear systematic code can be implemented easily. The encoding circuit is shown in Figure 3.2, where $\rightarrow\boxed{\ }\rightarrow$ denotes a shift-register stage (e.g., a flip-flop), $\oplus$ denotes a modulo-2 adder, and $\rightarrow(p_{ij})\rightarrow$ denotes a connection if $p_{ij} = 1$ and no connection if $p_{ij} = 0$. The encoding operation is very simple. The message $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ to be encoded is shifted into the message register and simultaneously into the channel. As soon as the entire message has entered the message register, the $n - k$ parity-check digits are formed at the outputs of the $n - k$ modulo-2 adders. These parity-check digits are then serialized and shifted into the channel. We see that the complexity of the encoding circuit is linearly proportional to the block length of the code. The encoding circuit for the (7, 4) code given in Table 3.1 is shown in Figure 3.3, where the connection is based on the parity-check equations given in Example 3.2.

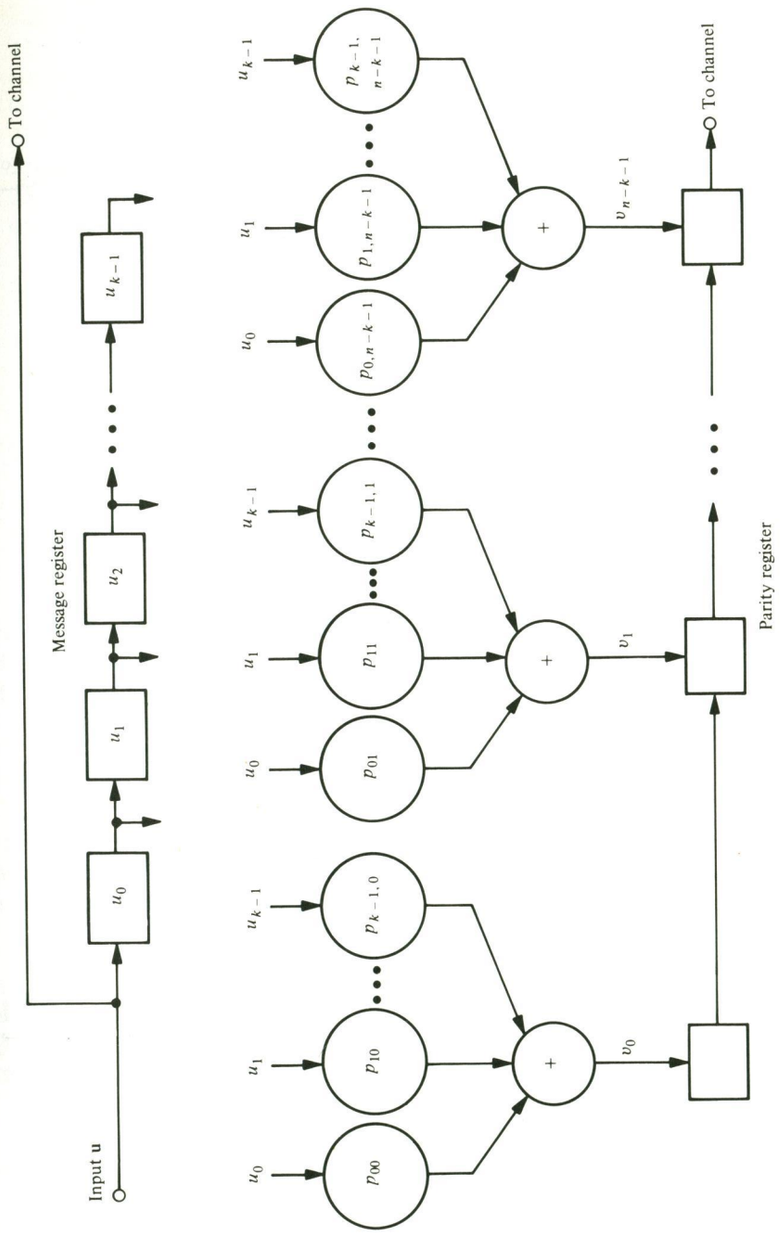

Figure 3.2 Encoding circuit for a linear systematic (n, k) code.

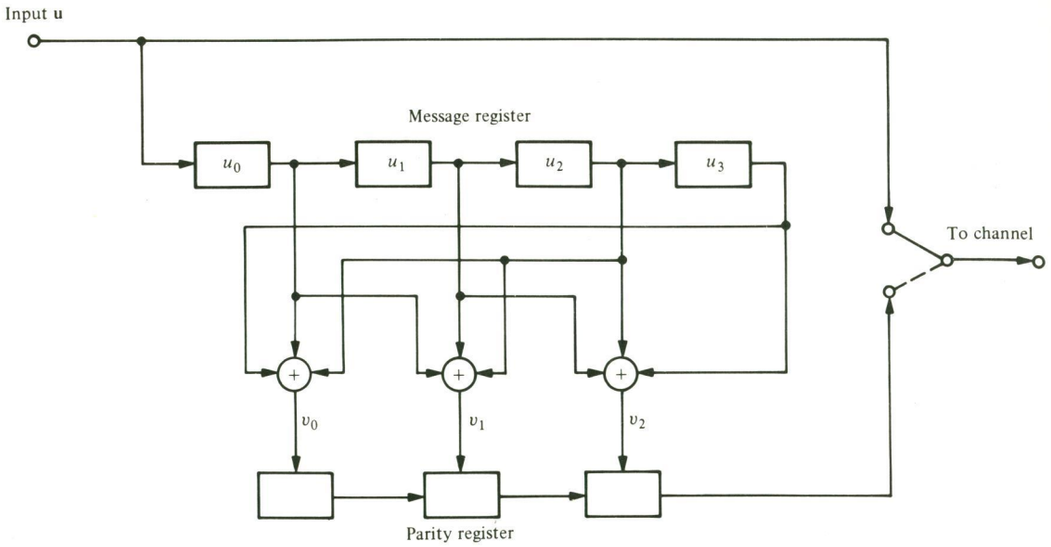

Figure 3.3 Encoding circuit for the (7, 4) systematic code given in Table 3.1.

3.2 SYNDROME AND ERROR DETECTION

Consider an (n, k) linear code with generator matrix $\mathbf{G}$ and parity-check matrix $\mathbf{H}$. Let $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be a code word that was transmitted over a noisy channel. Let $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1})$ be the received vector at the output of the channel. Because of the channel noise, $\mathbf{r}$ may be different from $\mathbf{v}$. The vector sum

$$\begin{aligned} \mathbf{e} &= \mathbf{r} + \mathbf{v} \\ &= (e_0, e_1, \ldots, e_{n-1}) \end{aligned} \tag{3.9}$$

is an n-tuple where $e_i = 1$ for $r_i \neq v_i$ and $e_i = 0$ for $r_i = v_i$. This n-tuple is called the *error vector* (or *error pattern*). The 1's in $\mathbf{e}$ are the *transmission errors* caused by the channel noise. It follows from (3.9) that the received vector $\mathbf{r}$ is the vector sum of the transmitted code word and the error vector, that is,

$$\mathbf{r} = \mathbf{v} + \mathbf{e}.$$

Of course, the receiver does not know either $\mathbf{v}$ or $\mathbf{e}$. Upon receiving $\mathbf{r}$, the decoder must first determine whether $\mathbf{r}$ contains transmission errors. If the presence of errors is detected, the decoder will either take actions to locate the errors and correct them (FEC) or request for a retransmission of $\mathbf{v}$(ARQ).

When $\mathbf{r}$ is received, the decoder computes the following $(n - k)$-tuple:

$$\begin{aligned} \mathbf{s} &= \mathbf{r} \cdot \mathbf{H}^T \\ &= (s_0, s_1, \ldots, s_{n-k}). \end{aligned} \tag{3.10}$$

which is called the *syndrome* of $\mathbf{r}$. Then $\mathbf{s} = \mathbf{0}$ if and only if $\mathbf{r}$ is a code word, and $\mathbf{s} \neq \mathbf{0}$

if and only if $\mathbf{r}$ is not a code word. Therefore, when $\mathbf{s} \neq \mathbf{0}$, we know that $\mathbf{r}$ is not a code word and the presence of errors has been detected. When $\mathbf{s} = \mathbf{0}$, $\mathbf{r}$ is a code word and the receiver accepts $\mathbf{r}$ as the transmitted code word. It is possible that the errors in certain error vectors are not detectable (i.e., $\mathbf{r}$ contains errors but $\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = \mathbf{0}$). This happens when the error pattern $\mathbf{e}$ is identical to a nonzero code word. In this event, $\mathbf{r}$ is the sum of two code words which is a code word, and consequently $\mathbf{r} \cdot \mathbf{H}^T = \mathbf{0}$. Error patterns of this kind are called *undetectable* error patterns. Since there are $2^k - 1$ nonzero code words, there are $2^k - 1$ undetectable error patterns. When an undetectable error pattern occurs, the decoder makes a *decoding error*. In a later section of the chapter we derive the probability of an undetected error for a BSC and show that this error probability can be made very small.

Based on (3.7) and (3.10), the syndrome digits are as follows:

$$\begin{aligned} s_0 &= r_0 + r_{n-k}p_{00} + r_{n-k+1}p_{10} + \cdots + r_{n-1}p_{k-1,0} \\ s_1 &= r_1 + r_{n-k}p_{01} + r_{n-k+1}p_{11} + \cdots + r_{n-1}p_{k-1,1} \\ &\cdots \\ s_{n-k-1} &= r_{n-k-1} + r_{n-k}p_{0,n-k-1} + r_{n-k+1}p_{1,n-k-1} + \cdots + r_{n-1}p_{k-1,n-k-1}. \end{aligned} \tag{3.11}$$

If we examine the equations above carefully, we find that the syndrome $\mathbf{s}$ is simply the vector sum of the received parity digits $(r_0, r_1, \ldots, r_{n-k-1})$ and the parity-check digits recomputed from the received information digits $r_{n-k}, r_{n-k+1}, \ldots, r_{n-1}$. Therefore, the syndrome can be formed by a circuit similar to the encoding circuit. A general syndrome circuit is shown in Figure 3.4.

Example 3.4

Consider the (7, 4) linear code whose parity-check matrix is given in Example 3.3. Let $\mathbf{r} = (r_0, r_1, r_2, r_3, r_4, r_5, r_6)$ be the received vector. Then the syndrome is given by

$$\begin{aligned} \mathbf{s} &= (s_0, s_1, s_2) \\ &= (r_0, r_1, r_2, r_3, r_4, r_5, r_6) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}. \end{aligned}$$

The syndrome digits are

$$\begin{aligned} s_0 &= r_0 + r_3 + r_5 + r_6 \\ s_1 &= r_1 + r_3 + r_4 + r_5 \\ s_2 &= r_2 + r_4 + r_5 + r_6. \end{aligned}$$

The syndrome circuit for this code is shown in Figure 3.5.

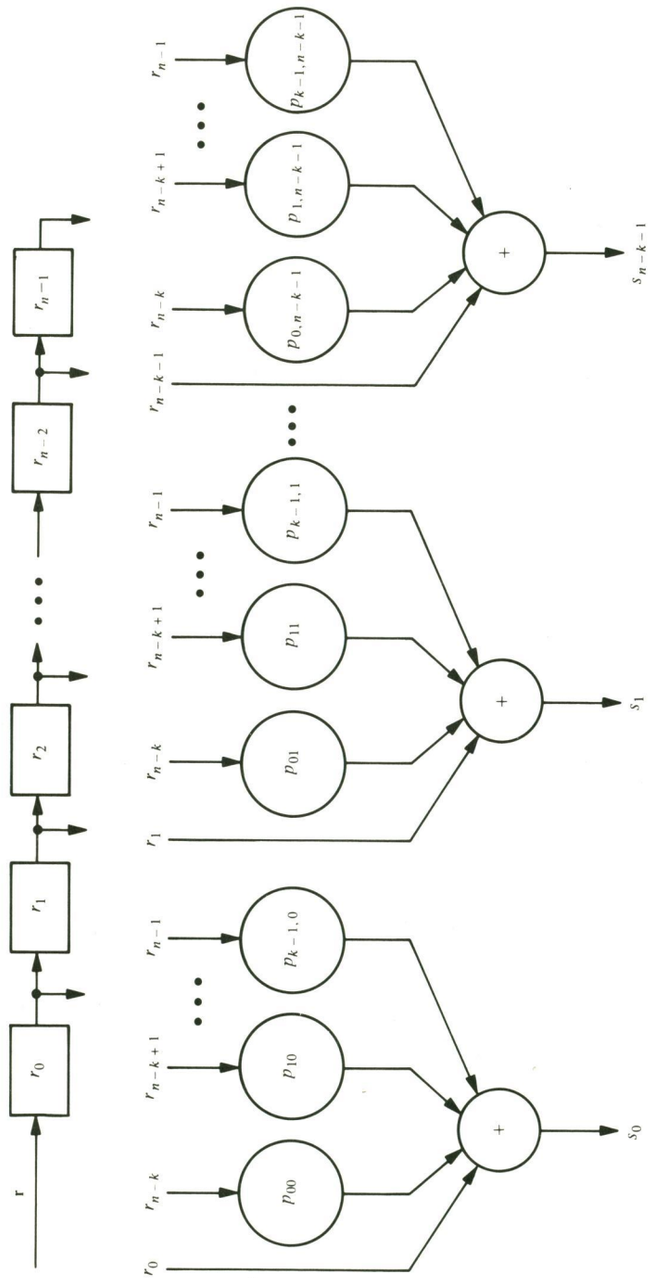

Figure 3.4 Syndrome circuit for a linear systematic (n, k) code.

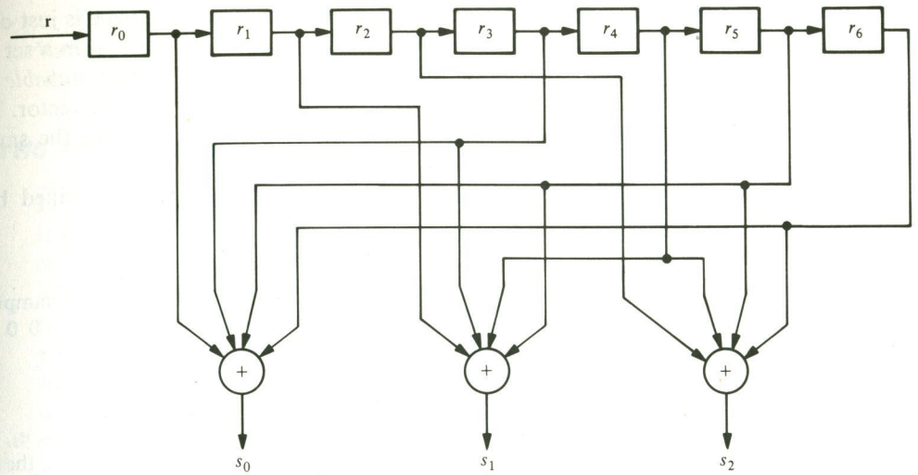

Figure 3.5 Syndrome circuit for the (7, 4) code given in Table 3.1.

The syndrome **s** computed from the received vector **r** actually depends only on the error pattern **e**, and not on the transmitted code word **v**. Since **r** is the vector sum of **v** and **e**, it follows from (3.10) that

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = (\mathbf{v} + \mathbf{e})\mathbf{H}^T = \mathbf{v} \cdot \mathbf{H}^T + \mathbf{e} \cdot \mathbf{H}^T.$$

However, $\mathbf{v} \cdot \mathbf{H}^T = \mathbf{0}$. Consequently, we obtain the following relation between the syndrome and the error pattern:

$$\mathbf{s} = \mathbf{e} \cdot \mathbf{H}^T. \tag{3.12}$$

If the parity-check matrix **H** is expressed in the systematic form as given by (3.7), multiplying out $\mathbf{e} \cdot \mathbf{H}^T$ yields the following linear relationship between the syndrome digits and the error digits:

$$\begin{aligned}
s_0 &= e_0 + e_{n-k}p_{00} + e_{n-k+1}p_{10} + \cdots + e_{n-1}p_{k-1,0} \\
s_1 &= e_1 + e_{n-k}p_{01} + e_{n-k+1}p_{11} + \cdots + e_{n-1}p_{k-1,1} \\
&\vdots \\
s_{n-k-1} &= e_{n-k-1} + e_{n-k}p_{0,n-k-1} + e_{n-k+1}p_{1,n-k-1} + \cdots + e_{n-1}p_{k-1,n-k-1}.
\end{aligned} \tag{3.13}$$

The syndrome digits are simply linear combinations of the error digits. Clearly, they provide information about the error digits and therefore can be used for error correction.

At this point, one would feel that any error correction scheme is a method of solving the $n - k$ linear equations of (3.13) for the error digits. Once the error pattern **e** has been found, the vector $\mathbf{r} + \mathbf{e}$ is taken as the actual transmitted code word. Unfortunately, determining the true error vector **e** is not a simple matter. This is because the $n - k$ linear equations of (3.13) do not have a unique solution but have 2^k solutions (this will be proved in Theorem 3.6). In other words, there are 2^k error

patterns that result in the same syndrome, and the true error pattern $\mathbf{e}$ is just one of them. Therefore, the decoder has to determine the true error vector from a set of 2^k candidates. To minimize the probability of a decoding error, the most *probable* error pattern that satisfies the equations of (3.13) is chosen as the true error vector. If the channel is a BSC, the most probable error pattern is the one that has the smallest number of nonzero digits.

The notion of using syndrome for error correction may be clarified by an example.

Example 3.5

Again, we consider the (7, 4) code whose parity-check matrix is given in Example 3.3. Let $\mathbf{v} = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ be the transmitted code word and $\mathbf{r} = (1\ 0\ 0\ 1\ 0\ 0\ 1)$ be the received vector. Upon receiving $\mathbf{r}$, the receiver computes the syndrome:

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = (1\ \ 1\ \ 1).$$

Next, the receiver attempts to determine the true error vector $\mathbf{e} = (e_0, e_1, e_2, e_3, e_4, e_5, e_6)$, which yields the syndrome above. It follows from (3.12) or (3.13) that the error digits are related to the syndrome digits by the following linear equations:

$$1 = e_0 + e_3 + e_5 + e_6$$
$$1 = e_1 + e_3 + e_4 + e_5$$
$$1 = e_2 + e_4 + e_5 + e_6.$$

There are $2^4 = 16$ error patterns that satisfy the equations above. They are

$$(0\ 0\ 0\ 0\ 0\ 1\ 0), \quad (1\ 0\ 1\ 0\ 0\ 1\ 1),$$
$$(1\ 1\ 0\ 1\ 0\ 1\ 0), \quad (0\ 1\ 1\ 1\ 0\ 1\ 1),$$
$$(0\ 1\ 1\ 0\ 1\ 1\ 0), \quad (1\ 1\ 0\ 0\ 1\ 1\ 1),$$
$$(1\ 0\ 1\ 1\ 1\ 1\ 0), \quad (0\ 0\ 0\ 1\ 1\ 1\ 1),$$
$$(1\ 1\ 1\ 0\ 0\ 0\ 0), \quad (0\ 1\ 0\ 0\ 0\ 0\ 1),$$
$$(0\ 0\ 1\ 1\ 0\ 0\ 0), \quad (1\ 0\ 0\ 1\ 0\ 0\ 1),$$
$$(1\ 0\ 0\ 0\ 1\ 0\ 0), \quad (0\ 0\ 1\ 0\ 1\ 0\ 1),$$
$$(0\ 1\ 0\ 1\ 1\ 0\ 0), \quad (1\ 1\ 1\ 1\ 1\ 0\ 1).$$

The error vector $\mathbf{e} = (0\ 0\ 0\ 0\ 0\ 1\ 0)$ has the smallest number of nonzero components. If the channel is a BSC, $\mathbf{e} = (0\ 0\ 0\ 0\ 0\ 1\ 0)$ is the most probable error vector that satisfies the equations above. Taking $\mathbf{e} = (0\ 0\ 0\ 0\ 0\ 1\ 0)$ as the true error vector, the receiver decodes the received vector $\mathbf{r} = (1\ 0\ 0\ 1\ 0\ 0\ 1)$ into the following code word:

$$\begin{aligned}\mathbf{v}^* &= \mathbf{r} + \mathbf{e} \\ &= (1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 1) + (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0) \\ &= (1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1).\end{aligned}$$

We see that $\mathbf{v}^*$ is the actual transmitted code word. Hence, the receiver has made a correct decoding. Later we show that the (7, 4) linear code considered in this example is capable of correcting any single error over a span of seven digits; that is, if a code word is transmitted and if only one digit is changed by the channel noise, the receiver will be able to determine the true error vector and to perform a correct decoding.

More discussion on error correction based on syndrome is given in Section 3.5. Various methods of determining the true error pattern from the $n - k$ linear equations of (3.13) are presented in later chapters.

3.3 THE MINIMUM DISTANCE OF A BLOCK CODE

In this section an important parameter of a block code called the *minimum distance* is introduced. This parameter determines the random-error-detecting and random-error-correcting capabilities of a code. Let $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be a binary n-tuple. The *Hamming weight* (or simply weight) of $\mathbf{v}$, denoted by $w(\mathbf{v})$, is defined as the number of nonzero components of $\mathbf{v}$. For example, the Hamming weight of $\mathbf{v} = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ is 4. Let $\mathbf{v}$ and $\mathbf{w}$ be two n-tuples. The *Hamming distance* (or simply distance) between $\mathbf{v}$ and $\mathbf{w}$, denoted $d(\mathbf{v}, \mathbf{w})$, is defined as the number of places where they differ. For example, the Hamming distance between $\mathbf{v} = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ and $\mathbf{w} = (0\ 1\ 0\ 0\ 0\ 1\ 1)$ is 3; they differ in the zeroth, first, and third places. The Hamming distance is a metric function that satisfies the *triangle inequality*. Let $\mathbf{v}$, $\mathbf{w}$, and $\mathbf{x}$ be three n-tuples. Then

$$d(\mathbf{v}, \mathbf{w}) + d(\mathbf{w}, \mathbf{x}) \geq d(\mathbf{v}, \mathbf{x}). \tag{3.14}$$

(The proof of this inequality is left as a problem.) It follows from the definition of Hamming distance and the definition of modulo-2 addition that the Hamming distance between two n-tuples, $\mathbf{v}$ and $\mathbf{w}$, is equal to the Hamming weight of the sum of $\mathbf{v}$ and $\mathbf{w}$, that is,

$$d(\mathbf{v}, \mathbf{w}) = w(\mathbf{v} + \mathbf{w}). \tag{3.15}$$

For example, the Hamming distance between $\mathbf{v} = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ and $\mathbf{w} = (1\ 1\ 1\ 0\ 0\ 1\ 0)$ is 4 and the weight of $\mathbf{v} + \mathbf{w} = (0\ 1\ 1\ 1\ 0\ 0\ 1)$ is also 4.

Given a block code C, one can compute the Hamming distance between any two distinct code words. The *minimum distance* of C, denoted $d_{\min}$, is defined as

$$d_{\min} = \min \{d(\mathbf{v}, \mathbf{w}) : \mathbf{v}, \mathbf{w} \in C, \mathbf{v} \neq \mathbf{w}\}. \tag{3.16}$$

If C is a linear block code, the sum of two vectors is also a code vector. It follows from (3.15) that the Hamming distance between two code vectors in C is equal to the Hamming weight of a third code vector in C. Then it follows from (3.16) that

$$\begin{aligned} d_{\min} &= \min \{w(\mathbf{v} + \mathbf{w}) : \mathbf{v}, \mathbf{w} \in C, \mathbf{v} \neq \mathbf{w}\} \\ &= \min \{w(\mathbf{x}) : \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\} \\ &\triangleq w_{\min}. \end{aligned} \tag{3.17}$$

The parameter $w_{\min} \triangleq \{w(\mathbf{x}) : \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\}$ is called the *minimum weight* of the linear code C. Summarizing the result above, we have the following theorem.

Theorem 3.1. The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words.

Therefore, for a linear block code, to determine the minimum distance of the code is equivalent to determine its minimum weight. The (7, 4) code given in Table 3.1 has minimum weight 3; thus, its minimum distance is 3. Next, we prove a number

of theorems that relate the weight structure of a linear block code to its parity-check matrix.

Theorem 3.2. Let C be an (n, k) linear code with parity-check matrix $\mathbf{H}$. For each code vector of Hamming weight l, there exist l columns of $\mathbf{H}$ such that the vector sum of these l columns is equal to the zero vector. Conversely, if there exist l columns of $\mathbf{H}$ whose vector sum is the zero vector, there exists a code vector of Hamming weight l in C.

Proof. Let us express the parity-check matrix in the following form:

$$\mathbf{H} = [\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{n-1}],$$

where $\mathbf{h}_i$ represents the ith column of $\mathbf{H}$. Let $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be a code vector of weight l. Then $\mathbf{v}$ has l nonzero components. Let $v_{i_1}, v_{i_2}, \ldots, v_{i_l}$ be the l nonzero components of $\mathbf{v}$, where $0 \leq i_1 < i_2 < \cdots < i_l \leq n-1$. Then $v_{i_1} = v_{i_2} = \cdots = v_{i_l} = 1$. Since $\mathbf{v}$ is a code vector, we must have

$$\begin{aligned}\mathbf{0} &= \mathbf{v} \cdot \mathbf{H}^T \\ &= v_0\mathbf{h}_0 + v_1\mathbf{h}_1 + \cdots + v_{n-1}\mathbf{h}_{n-1} \\ &= v_{i_1}\mathbf{h}_{i_1} + v_{i_2}\mathbf{h}_{i_2} + \cdots + v_{i_l}\mathbf{h}_{i_l} \\ &= \mathbf{h}_{i_1} + \mathbf{h}_{i_2} + \cdots + \mathbf{h}_{i_l}.\end{aligned}$$

This proves the first part of the theorem.

Now suppose that $\mathbf{h}_{i_1}, \mathbf{h}_{i_2}, \ldots, \mathbf{h}_{i_l}$ are l columns of $\mathbf{H}$ such that

$$\mathbf{h}_{i_1} + \mathbf{h}_{i_2} + \cdots + \mathbf{h}_{i_l} = \mathbf{0}. \tag{3.18}$$

Let us form a binary n-tuple $\mathbf{x} = (x_1, x_2, \ldots, x_{n-1})$ whose nonzero components are $x_{i_1}, x_{i_2}, \ldots, x_{i_l}$. The Hamming weight of $\mathbf{x}$ is l. Consider the product

$$\begin{aligned}\mathbf{x} \cdot \mathbf{H}^T &= x_0\mathbf{h}_0 + x_1\mathbf{h}_1 + \cdots + x_{n-1}\mathbf{h}_{n-1} \\ &= x_{i_1}\mathbf{h}_{i_1} + x_{i_2}\mathbf{h}_{i_2} + \cdots + x_{i_l}\mathbf{h}_{i_l} \\ &= \mathbf{h}_{i_1} + \mathbf{h}_{i_2} + \cdots + \mathbf{h}_{i_l}.\end{aligned}$$

It follows from (3.18) that $\mathbf{x} \cdot \mathbf{H}^T = \mathbf{0}$. Thus, $\mathbf{x}$ is a code vector of weight l in C. This proves the second part of the theorem. Q.E.D.

It follows from Theorem 3.2 that we have the following two corollaries.

Corollary 3.2.1. Let C be a linear block code with parity-check matrix $\mathbf{H}$. If no $d-1$ or fewer columns of $\mathbf{H}$ add to $\mathbf{0}$, the code has minimum weight at least d.

Corollary 3.2.2. Let C be a linear code with parity-check matrix $\mathbf{H}$. The minimum weight (or the minimum distance) of C is equal to the smallest number of columns of $\mathbf{H}$ that sum to $\mathbf{0}$.

Consider the (7, 4) linear code given in Table 3.1. The parity-check matrix of this code is

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

We see that all columns of $\mathbf{H}$ are nonzero and that no two of them are alike. Therefore, no two or fewer columns sum to $\mathbf{0}$. Hence, the minimum weight of this code is at least 3. However, the zeroth, second and sixth columns sum to $\mathbf{0}$. Thus, the minimum weight of the code is 3. From Table 3.1 we see that the minimum weight of the code is indeed 3. It follows from Theorem 3.1 that the minimum distance is 3.

Corollaries 3.2.1 and 3.2.2 are generally used to determine the minimum distance or to establish a lower bound on the minimum distance of a linear block code.

3.4 ERROR-DETECTING AND ERROR-CORRECTING CAPABILITIES OF A BLOCK CODE

When a code vector $\mathbf{v}$ is transmitted over a noisy channel, an error pattern of l errors will result in a received vector $\mathbf{r}$ which differs from the transmitted vector $\mathbf{v}$ in l places [i.e., $d(\mathbf{v}, \mathbf{r}) = l$]. If the minimum distance of a block code C is $d_{\min}$, any two distinct code vectors of C differ in at least $d_{\min}$ places. For this code C, no error pattern of $d_{\min} - 1$ or fewer errors can change one code vector into another. Therefore, any error pattern of $d_{\min} - 1$ or fewer errors will result in a received vector $\mathbf{r}$ that is not a code word in C. When the receiver detects that the received vector is not a code word of C, we say that errors are detected. Hence, a block code with minimum distance $d_{\min}$ is capable of detecting all the error patterns of $d_{\min} - 1$ or fewer errors. However, it cannot detect all the error patterns of $d_{\min}$ errors because there exists at least one pair of code vectors that differ in $d_{\min}$ places and there is an error pattern of $d_{\min}$ errors that will carry one into the other. The same argument applies to error patterns of more than $d_{\min}$ errors. For this reason, we say that the random-error-detecting capability of a block code with minimum distance $d_{\min}$ is $d_{\min} - 1$.

Even though a block code with minimum distance $d_{\min}$ guarantees detecting all the error patterns of $d_{\min} - 1$ or fewer errors, it is also capable of detecting a large fraction of error patterns with $d_{\min}$ or more errors. In fact, an (n, k) linear code is capable of detecting $2^n - 2^k$ error patterns of length n. This can be shown as follows. Among the $2^n - 1$ possible nonzero error patterns, there are $2^k - 1$ error patterns that are identical to the $2^k - 1$ nonzero code words. If any of these $2^k - 1$ error patterns occurs, it alters the transmitted code word $\mathbf{v}$ into another code word $\mathbf{w}$. Thus, $\mathbf{w}$ will be received and its syndrome is zero. In this case, the decoder accepts $\mathbf{w}$ as the transmitted code word and thus commits an incorrect decoding. Therefore, there are $2^k - 1$ *undetectable* error patterns. If an error pattern is not identical to a nonzero code word, the received vector $\mathbf{r}$ will not be a code word and the syndrome will not be zero. In this case, error will be detected. There are exactly $2^n - 2^k$ error patterns that are not identical to the code words of an (n, k) linear code. These $2^n - 2^k$ error patterns are *detectable* error patterns. For large n, $2^k - 1$ is in general much smaller than 2^n. Therefore, only a small fraction of error patterns pass through the decoder without being detected.

Let C be an (n, k) linear code. Let A_i be the number of code vectors of weight i in C. The numbers $A_0, A_1, \ldots, A_n$ are called the *weight distribution* of C. If C is used only for error detection on a BSC, the probability that the decoder fails to detect the presence of errors can be computed from the weight distribution of C. Let $P_u(E)$ denote the probability of an undetected error. Since an undetected error occurs only

when the error pattern is identical to a nonzero code vector of C,

$$P_u(E) = \sum_{i=1}^{n} A_i p^i (1-p)^{n-i}, \tag{3.19}$$

where p is the transition probability of the BSC. If the minimum distance of C is $d_{\min}$, then A_1 to $A_{d_{\min}-1}$ are zero.

Consider the (7, 4) code given in Table 3.1. The weight distribution of this code is $A_0 = 1$, $A_1 = A_2 = 0$, $A_3 = 7$, $A_4 = 7$, $A_5 = A_6 = 0$, and $A_7 = 1$. The probability of an undetected error is

$$P_u(E) = 7p^3(1-p)^4 + 7p^4(1-p)^3 + p^7.$$

If $p = 10^{-2}$, this probability is approximately 7×10^{-6}. In other words, if 1 million code words are transmitted over a BSC with $p = 10^{-2}$, there are on the average seven erroneous code words passing through the decoder without being detected.

If a block code C with minimum distance $d_{\min}$ is used for random-error correction, one would like to know how many errors that the code is able to correct. The minimum distance $d_{\min}$ is either odd or even. Let t be a positive integer such that

$$2t + 1 \leq d_{\min} \leq 2t + 2. \tag{3.20}$$

Next, we show that the code C is capable of correcting all the error patterns of t or fewer errors. Let $\mathbf{v}$ and $\mathbf{r}$ be the transmitted code vector and the received vector, respectively. Let $\mathbf{w}$ be any other code vector in C. The Hamming distances among $\mathbf{v}$, $\mathbf{r}$, and $\mathbf{w}$ satisfy the triangle inequality:

$$d(\mathbf{v}, \mathbf{r}) + d(\mathbf{w}, \mathbf{r}) \geq d(\mathbf{v}, \mathbf{w}). \tag{3.21}$$

Suppose that an error pattern of t' errors occurs during the transmission of $\mathbf{v}$. Then the received vector $\mathbf{r}$ differs from $\mathbf{v}$ in t' places and therefore $d(\mathbf{v}, \mathbf{r}) = t'$. Since $\mathbf{v}$ and $\mathbf{w}$ are code vectors in C, we have

$$d(\mathbf{v}, \mathbf{w}) \geq d_{\min} \geq 2t + 1. \tag{3.22}$$

Combining (3.21) and (3.22) and using the fact that $d(\mathbf{v}, \mathbf{r}) = t'$, we obtain the following inequality:

$$d(\mathbf{w}, \mathbf{r}) \geq 2t + 1 - t'.$$

If $t' \leq t$, then

$$d(\mathbf{w}, \mathbf{r}) > t.$$

The inequality above says that if an error pattern of t or fewer errors occurs, the received vector $\mathbf{r}$ is closer (in Hamming distance) to the transmitted code vector $\mathbf{v}$ than to any other code vector $\mathbf{w}$ in C. For a BSC, this means that the conditional probability $P(\mathbf{r} \mid \mathbf{v})$ is greater than the conditional probability $P(\mathbf{r} \mid \mathbf{w})$ for $\mathbf{w} \neq \mathbf{v}$. Based on the maximum likelihood decoding scheme, $\mathbf{r}$ is decoded into $\mathbf{v}$, which is the actual transmitted code vector. This results in a correct decoding and thus errors are corrected.

On the other hand, the code is not capable of correcting all the error patterns of l errors with $l > t$, for there is at least one case where an error pattern of l errors results in a received vector which is closer to an incorrect code vector than to the actual transmitted code vector. To show this, let $\mathbf{v}$ and $\mathbf{w}$ be two code vectors in C such that

$$d(\mathbf{v}, \mathbf{w}) = d_{\min}.$$

Let $\mathbf{e}_1$ and $\mathbf{e}_2$ be two error patterns that satisfy the following conditions:

(i) $\mathbf{e}_1 + \mathbf{e}_2 = \mathbf{v} + \mathbf{w}$.

(ii) $\mathbf{e}_1$ and $\mathbf{e}_2$ do not have nonzero components in common places.

Obviously, we have

$$w(\mathbf{e}_1) + w(\mathbf{e}_2) = w(\mathbf{v} + \mathbf{w}) = d(\mathbf{v}, \mathbf{w}) = d_{\min}. \tag{3.23}$$

Now suppose that $\mathbf{v}$ is transmitted and is corrupted by the error pattern $\mathbf{e}_1$. Then the received vector is

$$\mathbf{r} = \mathbf{v} + \mathbf{e}_1.$$

The Hamming distance between $\mathbf{v}$ and $\mathbf{r}$ is

$$d(\mathbf{v}, \mathbf{r}) = w(\mathbf{v} + \mathbf{r}) = w(\mathbf{e}_1). \tag{3.24}$$

The Hamming distance between $\mathbf{w}$ and $\mathbf{r}$ is

$$d(\mathbf{w}, \mathbf{r}) = w(\mathbf{w} + \mathbf{r}) = w(\mathbf{w} + \mathbf{v} + \mathbf{e}_1) = w(\mathbf{e}_2). \tag{3.25}$$

Now suppose that the error pattern $\mathbf{e}_1$ contains more than t errors [i.e., $w(\mathbf{e}_1) > t$]. Since $2t + 1 \leq d_{\min} \leq 2t + 2$, it follows from (3.23) that

$$w(\mathbf{e}_2) \leq t + 1.$$

Combining (3.24) and (3.25) and using the fact that $w(\mathbf{e}_1) > t$ and $w(\mathbf{e}_2) \leq t + 1$, we obtain the following inequality:

$$d(\mathbf{v}, \mathbf{r}) \geq d(\mathbf{w}, \mathbf{r}).$$

This inequality says that there exists an error pattern of l ($l > t$) errors which results in a received vector that is closer to an incorrect code vector than to the transmitted code vector. Based on the maximum likelihood decoding scheme, an incorrect decoding would be committed.

Summarizing the results above, a block code with minimum distance $d_{\min}$ *guarantees* correcting all the error patterns of $t = \lfloor (d_{\min} - 1)/2 \rfloor$ or fewer errors, where $\lfloor (d_{\min} - 1)/2 \rfloor$ denotes the largest integer no greater than $(d_{\min} - 1)/2$. The parameter $t = \lfloor (d_{\min} - 1)/2 \rfloor$ is called the *random-error-correcting capability* of the code. The code is referred to as a t-error-correcting code. The (7, 4) code given in Table 3.1 has minimum distance 3 and thus $t = 1$. It is capable of correcting any error pattern of single error over a block of seven digits.

A block code with random-error-correcting capability t is usually capable of correcting many error patterns of $t + 1$ or more errors. For a t-error-correcting (n, k) linear code, it is capable of correcting a total 2^{n-k} error patterns, including those with t or fewer errors (this will be seen in the next section). If a t-error-correcting block code is used strictly for error correction on a BSC with transition probability p, the probability that the decoder commits an erroneous decoding is upper bounded by

$$P(E) \leq \sum_{i=t+1}^{n} \binom{n}{i} p^i (1-p)^{n-i}. \tag{3.26}$$

In practice, a code is often used for correcting λ or fewer errors and simultaneously detecting l ($l > \lambda$) or fewer errors. That is, when λ or fewer errors occur, the

code is capable of correcting them; when more than λ but fewer than $l + 1$ errors occur, the code is capable of detecting their presence without making a decoding error. For this purpose, the minimum distance $d_{\min}$ of the code is at least $\lambda + l + 1$ (left as a problem). Thus, a block code with $d_{\min} = 10$ is capable of correcting three or fewer errors and simultaneously detecting six or fewer errors.

From the discussion above, we see that random-error-detecting and random-error-correcting capabilities of a block code are determined by the code's minimum distance. Clearly, for given n and k, one would like to construct a block code with minimum distance as large as possible, in addition to the implementation considerations.

3.5 STANDARD ARRAY AND SYNDROME DECODING

In this section a scheme for decoding linear block codes is presented. Let C be an (n, k) linear code. Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{2^k}$ be the code vectors of C. No matter which code vector is transmitted over a noisy channel, the received vector $\mathbf{r}$ may be any of the 2^n n-tuples over GF(2). Any decoding scheme used at the receiver is a rule to partition the 2^n possible received vectors into 2^k disjoint subsets $D_1, D_2, \ldots, D_{2^k}$ such that the code vector $\mathbf{v}_i$ is contained in the subset D_i for $1 \leq i \leq 2^k$. Thus, each subset D_i is one-to-one correspondence to a code vector $\mathbf{v}_i$. If the received vector $\mathbf{r}$ is found in the subset D_i, $\mathbf{r}$ is decoded into $\mathbf{v}_i$. Correct decoding is made if and only if the received vector $\mathbf{r}$ is in the subset D_i that corresponds to the actual code vector transmitted.

A method to partition the 2^n possible received vectors into 2^k disjoint subsets such that each subset contains one and only one code vector is described here. The partition is based on the linear structure of the code. First, the 2^k code vectors of C are placed in a row with the all-zero code vector $\mathbf{v}_1 = (0, 0, \ldots, 0)$ as the first (left-most) element. From the remaining $2^n - 2^k$ n-tuples, an n-tuple $\mathbf{e}_2$ is chosen and is placed under the zero vector $\mathbf{v}_1$. Now, we form a second row by adding $\mathbf{e}_2$ to each code vector $\mathbf{v}_i$ in the first row and placing the sum $\mathbf{e}_2 + \mathbf{v}_i$ under $\mathbf{v}_i$. Having completed the second row, an unused n-tuple $\mathbf{e}_3$ is chosen from the remaining n-tuples and is placed under $\mathbf{v}_1$. Then a third row is formed by adding $\mathbf{e}_3$ to each code vector $\mathbf{v}_i$ in the first row and placing $\mathbf{e}_3 + \mathbf{v}_i$ under $\mathbf{v}_i$. We continue this process until all the n-tuples are used. Then we have an array of rows and columns as shown in Figure 3.6. This array is called a *standard array* of the given linear code C.

It follows from the construction rule of a standard array that the sum of any two vectors in the same row is a code vector in C. Next, we prove some important properties of a standard array.

Theorem 3.3. No two n-tuples in the same row of a standard array are identical. Every n-tuple appears in one and only one row.

Proof. The first part of the theorem follows from the fact that all the code vectors of C are distinct. Suppose that two n-tuples in the lth rows are identical, say $\mathbf{e}_l + \mathbf{v}_i = \mathbf{e}_l + \mathbf{v}_j$ with $i \neq j$. This means that $\mathbf{v}_i = \mathbf{v}_j$, which is impossible. Therefore, no two n-tuples in the same row are identical.

It follows from the construction rule of the standard array that every n-tuple appears at least once. Now suppose that an n-tuple appears in both lth row and the

$$
\begin{array}{lllll}
\mathbf{v}_1 = 0 & \mathbf{v}_2 & \cdots & \mathbf{v}_i & \cdots & \mathbf{v}_{2^k} \\
\mathbf{e}_2 & \mathbf{e}_2 + \mathbf{v}_2 & \cdots & \mathbf{e}_2 + \mathbf{v}_i & \cdots & \mathbf{e}_2 + \mathbf{v}_{2^k} \\
\mathbf{e}_3 & \mathbf{e}_3 + \mathbf{v}_2 & \cdots & \mathbf{e}_3 + \mathbf{v}_i & \cdots & \mathbf{e}_3 + \mathbf{v}_{2^k} \\
\vdots & & & & & \vdots \\
\mathbf{e}_l & \mathbf{e}_l + \mathbf{v}_2 & \cdots & \mathbf{e}_l + \mathbf{v}_i & \cdots & \mathbf{e}_l + \mathbf{v}_{2^k} \\
\vdots & & & & & \vdots \\
\mathbf{e}_{2^{n-k}} & \mathbf{e}_{2^{n-k}} + \mathbf{v}_2 & \cdots & \mathbf{e}_{2^{n-k}} + \mathbf{v}_i & \cdots & \mathbf{e}_{2^{n-k}} + \mathbf{v}_{2^k}
\end{array}
$$

Figure 3.6 Standard array for an (n, k) linear code.

mth row with $l < m$. Then this n-tuple must be equal to $\mathbf{e}_l + \mathbf{v}_i$ for some i and equal to $\mathbf{e}_m + \mathbf{v}_j$ for some j. As a result, $\mathbf{e}_l + \mathbf{v}_i = \mathbf{e}_m + \mathbf{v}_j$. From this equality we obtain $\mathbf{e}_m = \mathbf{e}_l + (\mathbf{v}_i + \mathbf{v}_j)$. Since $\mathbf{v}_i$ and $\mathbf{v}_j$ are code vectors in C, $\mathbf{v}_i + \mathbf{v}_j$ is also a code vector in C, say $\mathbf{v}_s$. Then $\mathbf{e}_m = \mathbf{e}_l + \mathbf{v}_s$. This implies that the n-tuple $\mathbf{e}_m$ is in the lth row of the array, which contradicts the construction rule of the array that $\mathbf{e}_m$, the first element of the mth row, should be unused in any previous row. Therefore, no n-tuple can appear in more than one row of the array. This concludes the proof of the second part of the theorem. Q.E.D.

From Theorem 3.3 we see that there are $2^n/2^k = 2^{n-k}$ disjoint rows in the standard array, and that each row consists of 2^k distinct elements. The 2^{n-k} rows are called the *cosets* of the code C and the first n-tuple $\mathbf{e}_j$ of each coset is called a *coset leader*. Any element in a coset can be used as its coset leader. This does not change the elements of the coset; it simply permutes them.

Example 3.6

Consider the (6, 3) linear code generated by the following matrix:

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

The standard array of this code is shown in Figure 3.7.

A standard array of an (n, k) linear code C consists of 2^k disjoint columns. Each column consists of 2^{n-k} n-tuples with the topmost one as a code vector in C. Let D_j denote the jth column of the standard array. Then

$$
D_j = \{\mathbf{v}_j, \mathbf{e}_2 + \mathbf{v}_j, \mathbf{e}_3 + \mathbf{v}_j, \ldots, \mathbf{e}_{2^{n-k}} + \mathbf{v}_j\}, \tag{3.27}
$$

where $\mathbf{v}_j$ is a code vector of C and $\mathbf{e}_2, \mathbf{e}_3, \ldots, \mathbf{e}_{2^{n-k}}$ are the coset leaders. The 2^k disjoint columns $D_1, D_2, \ldots, D_{2^k}$ can be used for decoding the code C as described earlier in this section. Suppose that the code vector $\mathbf{v}_j$ is transmitted over a noisy channel. From (3.27) we see that the received vector $\mathbf{r}$ is in D_j if the error pattern caused by the channel is a coset leader. In this event, the received vector $\mathbf{r}$ will be decoded

Coset leader							
000000	001110	010101	100011	011011	101101	110110	111000
000001	001111	010100	100010	011010	101100	110111	111001
000010	001100	010111	100001	011001	101111	110100	111010
000100	001010	010001	100111	011111	101001	110010	111100
001000	000110	011101	101011	010011	100101	111110	110000
010000	011110	000101	110011	001011	111101	100110	101000
100000	101110	110101	000011	111011	001101	010110	011000
001001	000111	011100	101010	010010	100100	111111	110001

Figure 3.7 Standard array for the (6, 3) code.

correctly into the transmitted code vector $\mathbf{v}_j$. On the other hand, if the error pattern caused by the channel is not a coset leader, an erroneous decoding will result. This can be seen as follows. The error pattern $\mathbf{x}$ caused by the channel must be in some coset and under some nonzero code vector, say in the lth coset and under the code vector $\mathbf{v}_i \neq \mathbf{0}$. Then $\mathbf{x} = \mathbf{e}_l + \mathbf{v}_i$ and the received vector is

$$\mathbf{r} = \mathbf{v}_j + \mathbf{x} = \mathbf{e}_l + (\mathbf{v}_i + \mathbf{v}_j) = \mathbf{e}_l + \mathbf{v}_s.$$

The received vector $\mathbf{r}$ is thus in D_s and is decoded into $\mathbf{v}_s$, which is not the transmitted code vector. This results in an erroneous decoding. Therefore, the decoding is correct if and only if the error pattern caused by the channel is a coset leader. For this reason, the 2^{n-k} coset leaders (including the zero vector $\mathbf{0}$) are called the *correctable error patterns*. Summarizing the results above, we have the following theorem:

Theorem 3.4. Every (n, k) linear block code is capable of correcting 2^{n-k} error patterns.

To minimize the probability of a decoding error, the error patterns that are most likely to occur for a given channel should be chosen as the coset leaders. For a BSC, an error pattern of smaller weight is more probable than an error pattern of larger weight. Therefore, when a standard array is formed, each coset leader should be chosen to be a vector of *least weight* from the remaining available vectors. Choosing coset leaders in this manner, each coset leader has minimum weight in its coset. As a result, the decoding based on the standard array is the minimum distance decoding (i.e., the maximum likelihood decoding). To see this, let $\mathbf{r}$ be the received vector. Suppose that $\mathbf{r}$ is found in the ith column D_i and lth coset of the standard array. Then $\mathbf{r}$ is decoded into the code vector $\mathbf{v}_i$. Since $\mathbf{r} = \mathbf{e}_l + \mathbf{v}_i$, the distance between $\mathbf{r}$ and $\mathbf{v}_i$ is

$$d(\mathbf{r}, \mathbf{v}_i) = w(\mathbf{r} + \mathbf{v}_i) = w(\mathbf{e}_l + \mathbf{v}_i + \mathbf{v}_i) = w(\mathbf{e}_l). \tag{3.28}$$

Now, consider the distance between $\mathbf{r}$ and any other code vector, say $\mathbf{v}_j$,

$$d(\mathbf{r}, \mathbf{v}_j) = w(\mathbf{r} + \mathbf{v}_j) = w(\mathbf{e}_l + \mathbf{v}_i + \mathbf{v}_j).$$

Since $\mathbf{v}_i$ and $\mathbf{v}_j$ are two different code vectors, their vector sum, $\mathbf{v}_i + \mathbf{v}_j$, is a nonzero code vector, say $\mathbf{v}_s$. Thus,

$$d(\mathbf{r}, \mathbf{v}_j) = w(\mathbf{e}_l + \mathbf{v}_s). \tag{3.29}$$

Since $\mathbf{e}_l$ and $\mathbf{e}_l + \mathbf{v}_s$ are in the same coset and since $w(\mathbf{e}_l) \leq w(\mathbf{e}_l + \mathbf{v}_s)$, it follows from (3.28) and (3.29) that

$$d(\mathbf{r}, \mathbf{v}_i) \leq d(\mathbf{r}, \mathbf{v}_j).$$

This says that the received vector is decoded into a closest code vector. Hence, if each coset leader is chosen to have minimum weight in its coset, the decoding based on the standard array is the minimum distance decoding or MLD.

Let α_i denote the number of coset leaders of weight i. The numbers $\alpha_0, \alpha_1, \ldots, \alpha_n$ are called the *weight distribution* of the coset leaders. Knowing these numbers, we can compute the probability of a decoding error. Since a decoding error occurs if and only if the error pattern is not a coset leader, the error probability for a BSC with transition probability p is

$$P(E) = 1 - \sum_{i=0}^{n} \alpha_i p^i (1-p)^{n-i}. \tag{3.30}$$

Example 3.7

Consider the (6, 3) code given in Example 3.6. The standard array for this code is shown in Figure 3.7. The weight distribution of the coset leaders is $\alpha_0 = 1$, $\alpha_1 = 6$, $\alpha_2 = 1$, and $\alpha_3 = \alpha_4 = \alpha_5 = \alpha_6 = 0$. Thus,

$$P(E) = 1 - (1-p)^6 - 6p(1-p)^5 - p^2(1-p)^4.$$

For $p = 10^{-2}$, we have $P(E) \approx 1.37 \times 10^{-3}$.

An (n, k) linear code is capable of detecting $2^n - 2^k$ error patterns; however, it is capable of correcting only 2^{n-k} error patterns. For large n, 2^{n-k} is a small fraction of $2^n - 2^k$. Therefore, the probability of a decoding error is much higher than the probability of an undetected error.

Theorem 3.5. For an (n, k) linear code C with minimum distance $d_{\min}$, all the n-tuples of weight of $t = \lfloor (d_{\min} - 1)/2 \rfloor$ or less can be used as coset leaders of a standard array of C. If all the n-tuples of weight t or less are used as coset leaders, there is at least one n-tuple of weight $t + 1$ that cannot be used as a coset leader.

Proof. Since the minimum distance of C is $d_{\min}$, the minimum weight of C is also $d_{\min}$. Let $\mathbf{x}$ and $\mathbf{y}$ be two n-tuples of weight t or less. Clearly, the weight of $\mathbf{x} + \mathbf{y}$ is

$$w(\mathbf{x} + \mathbf{y}) \leq w(\mathbf{x}) + w(\mathbf{y}) \leq 2t < d_{\min}.$$

Suppose that $\mathbf{x}$ and $\mathbf{y}$ are in the same coset; then $\mathbf{x} + \mathbf{y}$ must be a nonzero code vector in C. This is impossible because the weight of $\mathbf{x} + \mathbf{y}$ is less than the minimum weight of C. Therefore, no two n-tuples of weight t or less can be in the same coset of C, and all the n-tuples of weight t or less can be used as coset leaders.

Let $\mathbf{v}$ be a minimum weight code vector of C [i.e., $w(\mathbf{v}) = d_{\min}$]. Let $\mathbf{x}$ and $\mathbf{y}$ be two n-tuples which satisfy the following two conditions:

(i) $\mathbf{x} + \mathbf{y} = \mathbf{v}$.
(ii) $\mathbf{x}$ and $\mathbf{y}$ do not have nonzero components in common places.

It follows from the definition that $\mathbf{x}$ and $\mathbf{y}$ must be in the same coset and

$$w(\mathbf{x}) + w(\mathbf{y}) = w(\mathbf{v}) = d_{\min}.$$

Suppose we choose $\mathbf{y}$ such that $w(\mathbf{y}) = t + 1$. Since $2t + 1 \leq d_{\min} \leq 2t + 2$, we have $w(\mathbf{x}) = t$ or $t + 1$. If $\mathbf{x}$ is used as a coset leader, then $\mathbf{y}$ cannot be a coset leader. Q.E.D.

Theorem 3.5 reconfirms the fact that an (n, k) linear code with minimum distance $d_{\min}$ is capable of correcting all the error patterns of $\lfloor (d_{\min} - 1)/2 \rfloor$ or fewer errors, but it is not capable of correcting all the error patterns of weight $t + 1$.

A standard array has an important property that can be used to simplify the decoding process. Let $\mathbf{H}$ be the parity-check matrix of the given (n, k) linear code C.

Theorem 3.6. All the 2^k n-tuples of a coset have the same syndrome. The syndromes for different cosets are different.

Proof. Consider the coset whose coset leader is $\mathbf{e}_l$. A vector in this coset is the sum of $\mathbf{e}_l$ and some code vector $\mathbf{v}_i$ in C. The syndrome of this vector is

$$(\mathbf{e}_l + \mathbf{v}_i)\mathbf{H}^T = \mathbf{e}_l\mathbf{H}^T + \mathbf{v}_i\mathbf{H}^T = \mathbf{e}_l\mathbf{H}^T$$

(since $\mathbf{v}_i\mathbf{H}^T = \mathbf{0}$). The equality above says that the syndrome of any vector in a coset is equal to the syndrome of the coset leader. Therefore, all the vectors of a coset have the same syndrome.

Let $\mathbf{e}_j$ and $\mathbf{e}_l$ be the coset leaders of the jth and lth cosets, respectively, where $j < l$. Suppose that the syndromes of these two cosets are equal. Then

$$\mathbf{e}_j\mathbf{H}^T = \mathbf{e}_l\mathbf{H}^T,$$
$$(\mathbf{e}_j + \mathbf{e}_l)\mathbf{H}^T = \mathbf{0}.$$

This implies that $\mathbf{e}_j + \mathbf{e}_l$ is a code vector in C, say $\mathbf{v}_i$. Thus, $\mathbf{e}_j + \mathbf{e}_l = \mathbf{v}_i$ and $\mathbf{e}_l = \mathbf{e}_j + \mathbf{v}_i$. This implies that $\mathbf{e}_l$ is in the jth coset, which contradicts the construction rule of a standard array that a coset leader should be previously unused. Therefore, no two cosets have the same syndrome. Q.E.D.

We recall that the syndrome of an n-tuple is an $(n - k)$-tuple and there are 2^{n-k} distinct $(n - k)$-tuples. It follows from Theorem 3.6 that there is a one-to-one correspondence between a coset and an $(n - k)$-tuple syndrome. Or, there is a one-to-one correspondence between a coset leader (a correctable error pattern) and a syndrome. Using this one-to-one correspondence relationship, we can form a decoding table, which is much simpler to use than a standard array. The table consists of 2^{n-k} coset leaders (the correctable error patterns) and their corresponding syndromes. This table is either stored or wired in the receiver. The decoding of a received vector consists of three steps:

Step 1. Compute the syndrome of $\mathbf{r}$, $\mathbf{r} \cdot \mathbf{H}^T$.

Step 2. Locate the coset leader $\mathbf{e}_l$ whose syndrome is equal to $\mathbf{r} \cdot \mathbf{H}^T$. Then $\mathbf{e}_l$ is assumed to be the error pattern caused by the channel.

Step 3. Decode the received vector $\mathbf{r}$ into the code vector $\mathbf{v} = \mathbf{r} + \mathbf{e}_l$.

The decoding scheme described above is called the *syndrome decoding* or *table-lookup decoding*. In principle, table-lookup decoding can be applied to any (n, k) linear code. It results in minimum decoding delay and minimum error probability. However,

for large $n - k$, the implementation of this decoding scheme becomes impractical, and either a large storage or a complicated logic circuitry is needed. Several practical decoding schemes which are variations of table-lookup decoding are discussed in subsequent chapters. Each of these decoding schemes requires additional properties in a code other than the linear structure.

Example 3.8

Consider the (7, 4) linear code given in Table 3.1. The parity-check matrix, as given in Example 3.3, is

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

The code has $2^3 = 8$ cosets and, therefore, there are eight correctable error patterns (including the all-zero vector). Since the minimum distance of the code is 3, it is capable of correcting all the error patterns of weight 1 or 0. Hence, all the 7-tuples of weight 1 or 0 can be used as coset leaders. There are $\binom{7}{0} + \binom{7}{1} = 8$ such vectors. We see that, for the (7, 4) linear code considered in this example, the number of correctable error patterns guaranteed by the minimum distance is equal to the total number of correctable error patterns. The correctable error patterns and their corresponding syndromes are given in Table 3.2.

TABLE 3.2 DECODING TABLE FOR THE (7, 4) LINEAR CODE GIVEN IN TABLE 3.1

Syndrome	Coset leaders
(1 0 0)	(1 0 0 0 0 0 0)
(0 1 0)	(0 1 0 0 0 0 0)
(0 0 1)	(0 0 1 0 0 0 0)
(1 1 0)	(0 0 0 1 0 0 0)
(0 1 1)	(0 0 0 0 1 0 0)
(1 1 1)	(0 0 0 0 0 1 0)
(1 0 1)	(0 0 0 0 0 0 1)

Suppose that the code vector $\mathbf{v} = (1\ 0\ 0\ 1\ 0\ 1\ 1)$ is transmitted and $\mathbf{r} = (1\ 0\ 0\ 1\ 1\ 1\ 1)$ is received. For decoding $\mathbf{r}$, we compute the syndrome of $\mathbf{r}$,

$$\mathbf{s} = (1\ 0\ 0\ 1\ 1\ 1\ 1) \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = (0\ 1\ 1).$$

From Table 3.2 we find that (0 1 1) is the syndrome of the coset leader $\mathbf{e} = (0\ 0\ 0\ 0\ 1\ 0\ 0)$. Thus, (0 0 0 0 1 0 0) is assumed to be the error pattern caused by the channel, and $\mathbf{r}$ is decoded into

$$\begin{aligned}\mathbf{v}^* &= \mathbf{r} + \mathbf{e} \\ &= (1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1) + (0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0) \\ &= (1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1),\end{aligned}$$

which is the actual code vector transmitted. The decoding is correct since the error pattern caused by the channel is a coset leader.

Now suppose that $\mathbf{v} = (0\ 0\ 0\ 0\ 0\ 0\ 0)$ is transmitted and $\mathbf{r} = (1\ 0\ 0\ 0\ 1\ 0\ 0)$ is received. We see that two errors have occurred during the transmission of $\mathbf{v}$. The error pattern is not correctable and will cause a decoding error. When $\mathbf{r}$ is received, the receiver computes the syndrome

$$\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = (1\ \ 1\ \ 1).$$

From the decoding table we find that the coset leader $\mathbf{e} = (0\ 0\ 0\ 0\ 0\ 1\ 0)$ corresponds to the syndrome $\mathbf{s} = (1\ 1\ 1)$. As a result, $\mathbf{r}$ is decoded into the code vector

$$\begin{aligned}\mathbf{v}^* &= \mathbf{r} + \mathbf{e} \\ &= (1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0) + (0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0) \\ &= (1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 0).\end{aligned}$$

Since $\mathbf{v}^*$ is not the actual code vector transmitted, a decoding error is committed.

Using Table 3.2, the code is capable of correcting any single error over a block of seven digits. When two or more errors occur, a decoding error will be committed.

The table-lookup decoding of an (n, k) linear code may be implemented as follows. The decoding table is regarded as the truth table of n switching functions:

$$\begin{aligned} e_0 &= f_0(s_0, s_1, \ldots, s_{n-k-1}), \\ e_1 &= f_1(s_0, s_1, \ldots, s_{n-k-1}), \\ &\vdots \\ e_{n-1} &= f_{n-1}(s_0, s_1, \ldots, s_{n-k-1}), \end{aligned}$$

where $s_0, s_1, \ldots, s_{n-k-1}$ are the syndrome digits, which are regarded as switching variables, and $e_0, e_1, \ldots, e_{n-1}$ are the estimated error digits. When these n switching functions are derived and simplified, a combinational logic circuit with the $n - k$ syndrome digits as inputs and the estimated error digits as outputs can be realized. The implementation of the syndrome circuit has been discussed in Section 3.2. The general decoder for an (n, k) linear code based on the table-lookup scheme is shown in Figure 3.8. The cost of this decoder depends primarily on the complexity of the combinational logic circuit.

Example 3.9

Again, we consider the (7, 4) code given in Table 3.1. The syndrome circuit for this code is shown in Figure 3.5. The decoding table is given by Table 3.2. From this table we form the truth table (Table 3.3). The switching expressions for the seven error digits are

$$\begin{aligned} e_0 &= s_0 \Lambda s_1' \Lambda s_2', & e_1 &= s_0' \Lambda s_1 \Lambda s_2', \\ e_2 &= s_0' \Lambda s_1' \Lambda s_2, & e_3 &= s_0 \Lambda s_1 \Lambda s_2', \\ e_4 &= s_0' \Lambda s_1 \Lambda s_2, & e_5 &= s_0 \Lambda s_1 \Lambda s_2, \\ e_6 &= s_0 \Lambda s_1' \Lambda s_2, \end{aligned}$$

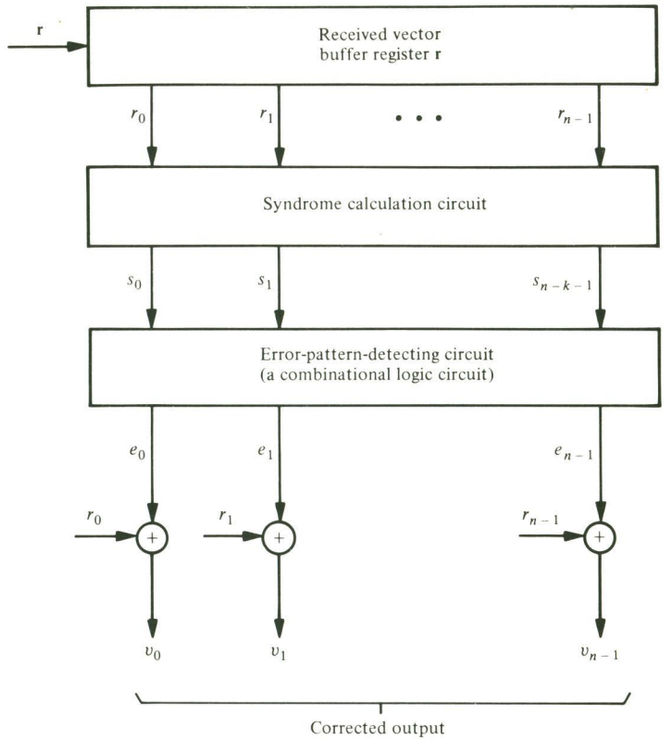

Figure 3.8 General decoder for a linear block code.

TABLE 3.3 TRUTH TABLE FOR THE ERROR DIGITS OF THE CORRECTABLE ERROR PATTERNS OF THE (7, 4) LINEAR CODE GIVEN IN TABLE 3.1

Syndromes			Correctable error patterns (coset leaders)						
s_0	s_1	s_2	e_0	e_1	e_2	e_3	e_4	e_5	e_6
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
1	1	0	0	0	0	1	0	0	0
0	1	1	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	1	0
1	0	1	0	0	0	0	0	0	1

where Λ denotes the logic-AND operation and s' denotes the logic-COMPLEMENT of s. These seven switching expressions can be realized by seven 3-input AND gates. The complete circuit of the decoder is shown in Figure 3.9.

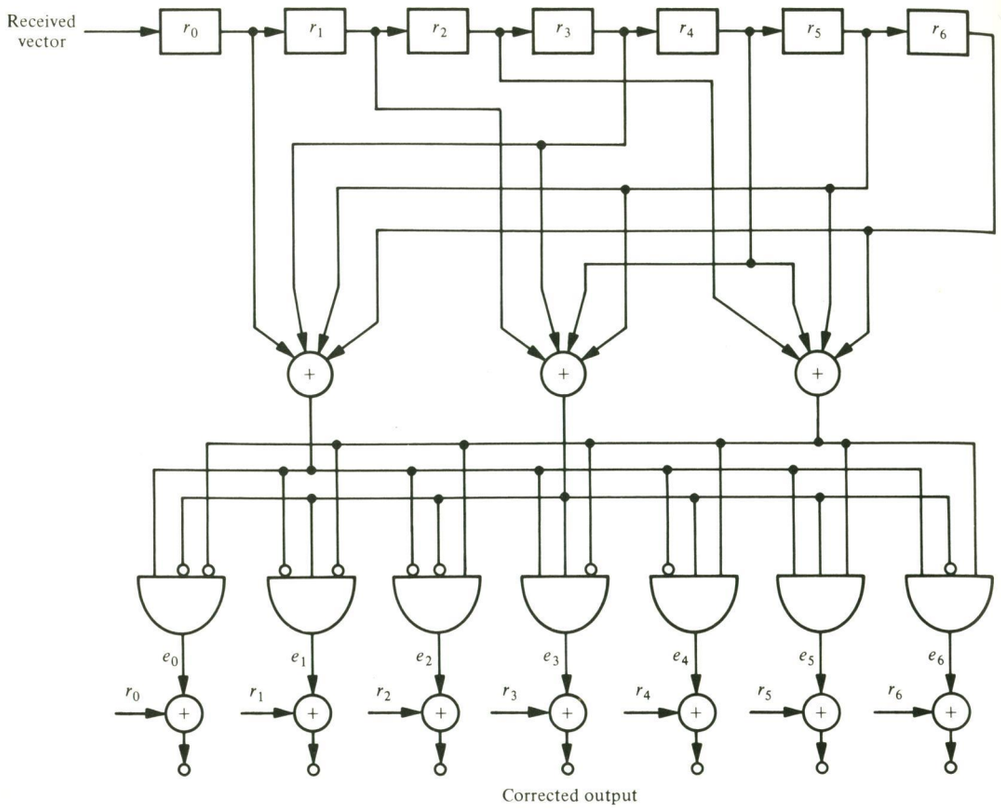

Figure 3.9 Decoding circuit for the (7, 4) given in Table 3.1.

3.6 PROBABILITY OF AN UNDETECTED ERROR FOR LINEAR CODES OVER A BSC

If an (n, k) linear code is used only for error detection over a BSC, the probability of an undetected error, $P_u(E)$, can be computed from (3.19) if the weight distribution of the code is known. There exists an interesting relationship between the weight distribution of a linear code and the weight distribution of its dual code. This relationship often makes the computation of $P_u(E)$ much easier. Let $\{A_0, A_1, \ldots, A_n\}$ be the weight distribution of an (n, k) linear code C and let $\{B_0, B_1, \ldots, B_n\}$ be the weight distribution of its dual code C_d. Now we represent these two weight distributions in polynomial form as follows:

$$\begin{aligned} A(z) &= A_0 + A_1 z + \cdots + A_n z^n, \\ B(z) &= B_0 + B_1 z + \cdots + B_n z^n. \end{aligned} \tag{3.31}$$

Then $A(z)$ and $B(z)$ are related by the following identity:

$$A(z) = 2^{-(n-k)}(1+z)^n B\left(\frac{1-z}{1+z}\right). \tag{3.32}$$

This identity is known as the *MacWilliams identity* [5]. The polynomials $A(z)$ and $B(z)$ are called the *weight enumerators* for the (n, k) linear code C and its dual C_d. From the MacWilliams identity, we see that if the weight distribution of the dual of a linear code is known, the weight distribution of the code itself can be determined. As a result, this gives us more flexibility of computing the weight distribution of a linear code.

Using the MacWilliams identity, we can compute the probability of an undetected error for an (n, k) linear code from the weight distribution of its dual. First, we put the expression of (3.19) into the following form:

$$\begin{aligned} P_u(E) &= \sum_{i=1}^{n} A_i p^i (1-p)^{n-i} \\ &= (1-p)^n \sum_{i=1}^{n} A_i \left(\frac{p}{1-p}\right)^i. \end{aligned} \tag{3.33}$$

Substituting $z = p/(1-p)$ in $A(z)$ of (3.31) and using the fact that $A_0 = 1$, we obtain the following identity:

$$A\left(\frac{p}{1-p}\right) - 1 = \sum_{i=1}^{n} A_i \left(\frac{p}{1-p}\right)^i. \tag{3.34}$$

Combining (3.33) and (3.34), we have the following expression for the probability of an undetected error:

$$P_u(E) = (1-p)^n \left[A\left(\frac{p}{1-p}\right) - 1 \right]. \tag{3.35}$$

From (3.35) and the MacWilliams identity of (3.32), we finally obtain the following expression for $P_u(E)$:

$$P_u(E) = 2^{-(n-k)} B(1-2p) - (1-p)^n, \tag{3.36}$$

where

$$B(1-2p) = \sum_{i=0}^{n} B_i (1-2p)^i.$$

Hence, there are two ways for computing the probability of an undetected error for a linear code; often one is easier than the other. If $n - k$ is smaller than k, it is much easier to compute $P_u(E)$ from (3.36); otherwise, it is easier to use (3.35).

Example 3.10

Consider the (7, 4) linear code given in Table 3.1. The dual of this code is generated by its parity-check matrix,

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

(see Example 3.3). Taking the linear combinations of the rows of **H**, we obtain the following eight vectors in the dual code:

(0 0 0 0 0 0 0), (1 1 0 0 1 0 1),
(1 0 0 1 0 1 1), (1 0 1 1 1 0 0),
(0 1 0 1 1 1 0), (0 1 1 1 0 0 1),
(0 0 1 0 1 1 1), (1 1 1 0 0 1 0).

Thus, the weight enumerator for the dual code is $B(z) = 1 + 7z^4$. Using (3.36), we obtain the probability of an undetected error for the (7, 4) linear code given in Table 3.1,

$$P_u(E) = 2^{-3}[1 + 7(1 - 2p)^4] - (1 - p)^7.$$

This probability was also computed in Section 3.4 using the weight distribution of the code itself.

Theoretically, we can compute the weight distribution of an (n, k) linear code by examining its 2^k code words or by examining the 2^{n-k} code words of its dual and then applying the MacWilliams identity. However, for large n, k, and $n - k$, the computation becomes practically impossible. Except for some short linear codes and a few small classes of linear codes, the weight distributions for many known linear codes are still unknown. Consequently, it is very difficult, if not impossible, to compute their probability of an undetected error.

Although it is difficult to compute the probability of an undetected error for a specific (n, k) linear code for large n and k, it is quite easy to derive an upper bound on the average probability of an undetected error for the ensemble of all (n, k) linear systematic codes. As we have shown earlier, an (n, k) linear systematic code is completely specified by a matrix $\mathbf{G}$ of the form given by (3.4). The submatrix $\mathbf{P}$ consists of $k(n - k)$ entries. Since each entry p_{ij} can be either a 0 or a 1, there are $2^{k(n-k)}$ distinct matrices $\mathbf{G}$'s of the form given by (3.4). Let Γ denote the ensemble of codes generated by these $2^{k(n-k)}$ matrices. Suppose that we choose a code randomly from Γ and use it for error detection. Let C_j be the chosen code. Then the probability of C_j being chosen is

$$P(C_j) = 2^{-k(n-k)}. \tag{3.37}$$

Let A_{ji} denote the number of code words in C_j with weight i. It follows from (3.19) that probability of an undetected error for C_j is given by

$$P_u(E|C_j) = \sum_{i=1}^{n} A_{ji} p^i (1 - p)^{n-i}. \tag{3.38}$$

The average probability of an undetected error for a linear code in Γ is defined as

$$\mathbf{P}_u(\mathbf{E}) = \sum_{j=1}^{|\Gamma|} P(C_j) P_u(E|C_j), \tag{3.39}$$

where $|\Gamma|$ denotes the number of codes in Γ. Substituting (3.37) and (3.38) into (3.39), we obtain

$$\mathbf{P}_u(\mathbf{E}) = 2^{-k(n-k)} \sum_{i=1}^{n} p^i (1 - p)^{n-i} \sum_{j=1}^{|\Gamma|} A_{ji}. \tag{3.40}$$

A nonzero n-tuple is either contained in exactly $2^{(k-1)(n-k)}$ codes in Γ or contained in none of the codes (left as a problem). Since there are $\binom{n}{i}$ n-tuples of weight i, we have

$$\sum_{j=1}^{|\Gamma|} A_{ji} \leq \binom{n}{i} 2^{(k-1)(n-k)}. \tag{3.41}$$

Substituting (3.41) into (3.40), we obtain the following upper bound on the average probability of an undetected error for an (n, k) linear systematic code:

$$\begin{aligned} \mathbf{P}_u(\mathbf{E}) &\leq 2^{-(n-k)} \sum_{i=1}^{n} \binom{n}{i} p^i (1 - p)^{n-i} \\ &= 2^{-(n-k)}[1 - (1 - p)^n]. \end{aligned} \tag{3.42}$$

Since $[1 - (1 - p)^n] \leq 1$, it is clear that $\mathbf{P}_u(\mathbf{E}) \leq 2^{-(n-k)}$.

The result above says that there exist (n, k) linear codes with probability of an undetected error, $P_u(E)$, upper bounded by $2^{-(n-k)}$. In other words, there exist (n, k) linear codes with $P_u(E)$ *decreasing exponentially with the number of parity-check digits*, $n - k$. Even for moderate $n - k$, these codes have a very small probability of an undetected error. For example, let $n - k = 30$. There exist (n, k) linear codes for which $P_u(E)$ is upper bounded by $2^{-30} \approx 10^{-9}$. Many classes of linear codes have been constructed for the past three decades. However, only a few small classes of linear codes have been proved to have $P_u(E)$ satisfying the upper bound $2^{-(n-k)}$. It is still not known whether the other known linear codes satisfy this upper bound. A class of linear codes that satisfies this upper bound is presented in the next section. Other codes with probability of an undetected error decreasing exponentially with $n - k$ are presented in subsequent chapters.

3.7 HAMMING CODES

Hamming codes are the first class of linear codes devised for error correction [6]. These codes and their variations have been widely used for error control in digital communication and data storage systems.

For any positive integer $m \geq 3$, there exists a Hamming code with the following parameters:

Code length:	$n = 2^m - 1$
Number of information symbols:	$k = 2^m - m - 1$
Number of parity-check symbols:	$n - k = m$
Error-correcting capability:	$t = 1 (d_{\min} = 3)$.

The parity-check matrix $\mathbf{H}$ of this code consists of all the nonzero m-tuples as its columns. In systematic form, the columns of $\mathbf{H}$ are arranged in the following form:

$$\mathbf{H} = [\mathbf{I}_m \quad \mathbf{Q}],$$

where $\mathbf{I}_m$ is an $m \times m$ identity matrix and the submatrix $\mathbf{Q}$ consists of $2^m - m - 1$ columns which are the m-tuples of weight 2 or more. For example, let $m = 3$. The parity-check matrix of a Hamming code of length 7 can be put in the form

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

which is the parity-check matrix of the (7, 4) linear code given in Table 3.1 (see Example 3.3). Hence, the code given in Table 3.1 is a Hamming code. The columns of $\mathbf{Q}$ may be arranged in any order without affecting the distance property and weight

distribution of the code. In systematic form, the generator matrix of the code is

$$\mathbf{G} = [\mathbf{Q}^T \quad \mathbf{I}_{2^m-m-1}],$$

where $\mathbf{Q}^T$ is the transpose of $\mathbf{Q}$ and $\mathbf{I}_{2^m-m-1}$ is an $(2^m - m - 1) \times (2^m - m - 1)$ identity matrix.

Since the columns of $\mathbf{H}$ are nonzero and distinct, no two columns add to zero. It follows from Corollary 3.2.1 that the minimum distance of a Hamming code is at least 3. Since $\mathbf{H}$ consists of all the nonzero m-tuples as its columns, the vector sum of any two columns, say $\mathbf{h}_i$ and $\mathbf{h}_j$, must also be a column in $\mathbf{H}$, say $\mathbf{h}_l$. Thus,

$$\mathbf{h}_i + \mathbf{h}_j + \mathbf{h}_l = \mathbf{0}.$$

It follows from Corollary 3.2.2 that the minimum distance of a Hamming code is exactly 3. Hence, the code is capable of correcting all the error patterns of single error or of detecting all the error patterns of two or fewer errors.

If we form the standard array for the Hamming code of length $2^m - 1$, all the $(2^m - 1)$-tuples of weight 1 can be used as coset leaders (Theorem 3.5). The number of $(2^m - 1)$-tuples of weight 1 is $2^m - 1$. Since $n - k = m$, the code has 2^m cosets. Thus, the zero vector $\mathbf{0}$ and the $(2^m - 1)$-tuples of weight 1 form all the coset leaders of the standard array. This says that a Hamming code corrects only the error patterns of single error and no others. This is a very interesting structure. A t-error-correcting code is called a *perfect code* if its standard array has all the error patterns of t or fewer errors and no others as coset leaders. Thus, Hamming codes form a class of single-error-correcting perfect codes. Perfect codes are rare [3]. Besides the Hamming codes, the only other nontrivial binary perfect code is the (23, 12) Golay code (see Section 5.3).

Decoding of Hamming codes can be accomplished easily with the table-lookup scheme described in Section 3.5. The decoder for a Hamming code of length $2^m - 1$ can be implemented in the same manner as that for the (7,4) Hamming code given in Example 3.9.

We may delete any l columns from the parity-check matrix $\mathbf{H}$ of a Hamming code. This deletion results in an $m \times (2^m - l - 1)$ matrix $\mathbf{H}'$. Using $\mathbf{H}'$ as a parity-check matrix, we obtain a shortened Hamming code with the following parameters:

Code length:	$n = 2^m - l - 1$
Number of information symbols:	$k = 2^m - m - l - 1$
Number of parity-check symbols:	$n - k = m$
Minimum distance:	$d_{\min} \geq 3$.

If we delete columns from $\mathbf{H}$ properly, we may obtain a shortened Hamming code with minimum distance 4. For example, if we delete from the submatrix $\mathbf{Q}$ all the columns of even weight, we obtain an $m \times 2^{m-1}$ matrix

$$\mathbf{H}' = [\mathbf{I}_m \quad \mathbf{Q}'],$$

where $\mathbf{Q}'$ consists of $2^{m-1} - m$ columns of odd weight. Since all the columns of $\mathbf{H}'$ have odd weight, no three columns add to zero. However, for a column $\mathbf{h}_i$ of weight 3 in $\mathbf{Q}'$, there exists three columns $\mathbf{h}_j$, $\mathbf{h}_l$, and $\mathbf{h}_s$ in $\mathbf{I}_m$ such that $\mathbf{h}_i + \mathbf{h}_j + \mathbf{h}_l + \mathbf{h}_s = \mathbf{0}$.

Thus, the shortened Hamming code with $\mathbf{H}'$ as a parity-check matrix has minimum distance exactly 4.

The distance —4 shortened Hamming code can be used for correcting all error patterns of single error and simultaneously detecting all error patterns of double errors. When a single error occurs during the transmission of a code vector, the resultant syndrome is nonzero and it contains an odd number of 1's. However, when double errors occur, the syndrome is also nonzero, but it contains even number of 1's. Based on these facts, decoding can be accomplished in the following manner:

1. If the syndrome $\mathbf{s}$ is zero, we assume that no error occurred.
2. If $\mathbf{s}$ is nonzero and it contains odd number of 1's, we assume that a single error occurred. The error pattern of a single error that corresponds to $\mathbf{s}$ is added to the received vector for error correction.
3. If $\mathbf{s}$ is nonzero and it contains even number of 1's, an uncorrectable error pattern has been detected.

A class of single-error-correcting and double-error-detecting shortened Hamming codes which is widely used for error control in computer main/or control storages is presented in Chapter 16.

The weight distribution of a Hamming code of length $n = 2^m - 1$ is known [1–4]. The number of code vectors of weight i, A_i, is simply the coefficient of z^i in the expansion of the following polynomial:

$$A(z) = \frac{1}{n+1}\{(1+z)^n + n(1-z)(1-z^2)^{(n-1)/2}\}. \tag{3.43}$$

This polynomial is the weight enumerator for the Hamming codes.

Example 3.11

Let $m = 3$. Then $n = 2^3 - 1 = 7$ and the weight enumerator for the (7, 4) Hamming code is

$$A(z) = \tfrac{1}{8}\{(1+z)^7 + 7(1-z)(1-z^2)^3\} = 1 + 7z^3 + 7z^4 + z^7.$$

Hence, the weight distribution for the (7, 4) Hamming code is $A_0 = 1$, $A_3 = A_4 = 7$, and $A_7 = 1$.

The dual code of a $(2^m - 1, 2^m - m - 1)$ Hamming code is a $(2^m - 1, m)$ linear code. This code has a very simple weight distribution; it consists of the all-zero code word and $2^m - 1$ code words of weight 2^{m-1}. Thus, its weight enumerator is

$$B(z) = 1 + (2^m - 1)z^{2^{m-1}}. \tag{3.44}$$

The duals of Hamming codes are discussed further in Chapter 7.

If a Hamming code is used for error detection over a BSC, its probability of an undetected error, $P_u(E)$, can be computed either from (3.35) and (3.43) or from (3.36) and (3.44). Computing $P_u(E)$ from (3.36) and (3.44) is easier. Combining (3.36) and (3.44), we obtain

$$P_u(E) = 2^{-m}\{1 + (2^m - 1)(1-2p)^{2^{m-1}}\} - (1-p)^{2^{m-1}}. \tag{3.45}$$

The probability $P_u(E)$ for Hamming codes does satisfy the upper bound $2^{-(n-k)} = 2^{-m}$ for $p \leq \frac{1}{2}$ [i.e., $P_u(E) \leq 2^{-m}$] [7]. This can be shown by using the expression of (3.45) (see Problem 3.21).

PROBLEMS

3.1. Consider a systematic (8, 4) code whose parity-check equations are

$$v_0 = u_1 + u_2 + u_3,$$
$$v_1 = u_0 + u_1 + u_2,$$
$$v_2 = u_0 + u_1 + u_3,$$
$$v_3 = u_0 + u_2 + u_3.$$

where u_0, u_1, u_2, and u_3 are message digits and v_0, v_1, v_2, and v_3 are parity-check digits. Find the generator and parity-check matrices for this code. Show analytically that the minimum distance of this code is 4.

3.2. Construct an encoder for the code given in Problem 3.1.

3.3. Construct a syndrome circuit for the code given in Problem 3.1.

3.4. Let $\mathbf{H}$ be the parity-check matrix of an (n, k) linear code C that has both odd- and even-weight code vectors. Construct a new linear code C_1 with the following parity-check matrix:

$$\mathbf{H}_1 = \begin{bmatrix} 0 & & \\ 0 & & \\ \cdot & & \\ \cdot & \mathbf{H} & \\ \cdot & & \\ 0 & & \\ \hline 1 & 1\ 1\ \cdots\ 1 & \end{bmatrix}.$$

(Note that the last row of $\mathbf{H}_1$ consists of all 1's)

(a) Show that C_1 is an $(n + 1, k)$ linear code. C_1 is called an *extension* of C.

(b) Show that every code vector of C_1 has even wight.

(c) Show that C_1 can be obtained from C by adding an extra parity-check digit, denoted v_∞, to the left of each code vector $\mathbf{v}$ as follows: (1) if $\mathbf{v}$ has odd weight, then $v_\infty = 1$, and (2) if $\mathbf{v}$ has even weight, then $v_\infty = 0$. The parity-check digit v_∞ is called an *overall parity-check* digit.

3.5. Let C be a linear code with both even-weight and odd-weight code vectors. Show that the number of even-weight code vectors is equal to the number of odd-weight code vectors.

3.6. Consider an (n, k) linear code C whose generator matrix $\mathbf{G}$ contains no zero column. Arrange all the code vectors of C as rows of a 2^k-by-n array.

(a) Show that no column of the array contains only zeros.

(b) Show that each column of the array consists of 2^{k-1} zeros and 2^{k-1} ones.

(c) Show that the set of all code vectors with zeros in a particular component forms a subspace of C. What is the dimension of this subspace?

3.7. Prove that the Hamming distance satisfies the triangle inequality; that is, let $\mathbf{x}$, $\mathbf{y}$, and $\mathbf{z}$ be three n-tuples over GF(2), and show that

$$d(\mathbf{x}, \mathbf{y}) + d(\mathbf{y}, \mathbf{z}) \geq d(\mathbf{x}, \mathbf{z}).$$

3.8. Prove that a linear code is capable of correcting λ or fewer errors and simultaneously detecting $l (l > \lambda)$ or fewer errors if its minimum distance $d_{\min} \geq \lambda + l + 1$.

3.9. Determine the weight distribution of the (8, 4) linear code given in Problem 3.1. Let the transition probability of a BSC be $p = 10^{-2}$. Compute the probability of an undetected error of this code.

3.10. Since the (8, 4) linear code given in Problem 3.1 has minimum distance 4, it is capable of correcting all the single-error patterns and simultaneously detecting any combination of double errors. Construct a decoder for this code. The decoder must be capable of correcting any single error and detecting any double errors.

3.11. Let Γ be the ensemble of all the binary systematic (n, k) linear codes. Prove that a nonzero binary n-tuple $\mathbf{v}$ is either contained in exactly $2^{(k-1)(n-k)}$ codes in Γ or contained in none of the codes in Γ.

3.12. The (8, 4) linear code given in Problem 3.1 is capable of correcting 16 error patterns (the coset leaders of a standard array). Suppose that this code is used for a BSC. Devise a decoder for this code based on the table-lookup decoding scheme. The decoder is designed to correct the 16 most probable error patterns.

3.13. Let C_1 be an (n_1, k) linear systematic code with minimum distance d_1 and generator matrix $\mathbf{G}_1 = [\mathbf{P}_1\ \mathbf{I}_k]$. Let C_2 be an (n_2, k) linear systematic code with minimum distance d_2 and generator matrix $\mathbf{G}_2 = [\mathbf{P}_2\ \mathbf{I}_k]$. Consider an $(n_1 + n_2, k)$ linear code with the following parity-check matrix:

$$\mathbf{H} = \begin{bmatrix} & \mathbf{P}_1^T \\ \mathbf{I}_{n_1+n_2-k} & \mathbf{I}_k \\ & \mathbf{P}_2^T \end{bmatrix}.$$

Show that this code has minimum distance at least $d_1 + d_2$.

3.14. Show that the dual code of the (8, 4) linear code C given in Problem 3.1 is identical to C. C is said to be *self-dual.*

3.15. Form a parity-check matrix for a (15, 11) Hamming code. Devise a decoder for this code.

3.16. For any binary (n, k) linear code with minimum distance (or minimum weight) $2t + 1$ or greater, the number of parity-check digits satisfies the following inequality:

$$n - k \geq \log_2 \left[1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}\right].$$

The inequality above gives an upper bound on the random error-correcting capability t of an (n, k) linear code. This bound is known as the *Hamming bound* [5]. [*Hint:* For an (n, k) linear code with minimum distance $2t + 1$ or greater, all the n-tuples of weight t or less can be used as coset leaders in a standard array.]

3.17. Show that the Hamming codes achieve the Hamming bound.

3.18. Show that the minimum distance $d_{\min}$ of an (n, k) linear code satisfies the following inequality:

$$d_{\min} \leq \frac{n \cdot 2^{k-1}}{2^k - 1}.$$

(*Hint:* Use the result of Problem 3.6(b). The bound above is known as the *Plotkin bound* [1–3].)

3.19. Show that there exists an (n, k) linear code with minimum distance at least d if

$$\sum_{i=1}^{d-1} \binom{n}{i} < 2^{n-k}.$$