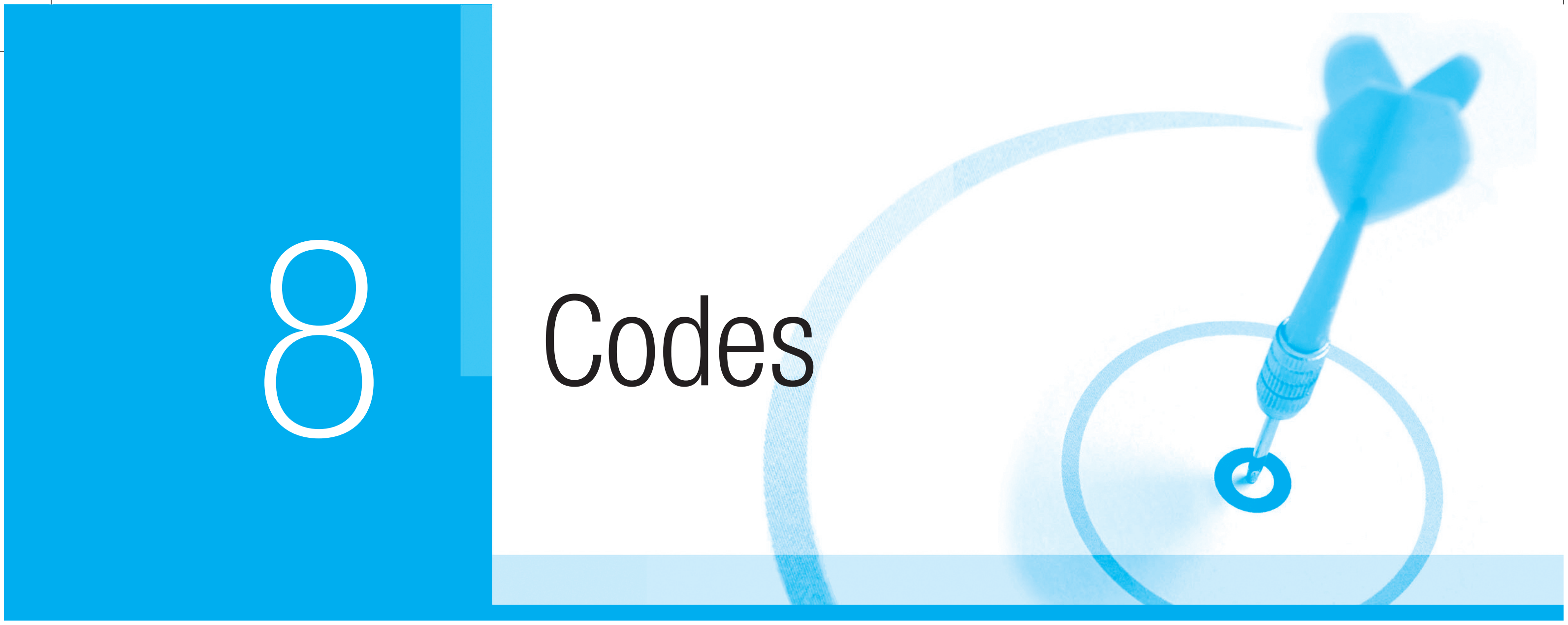
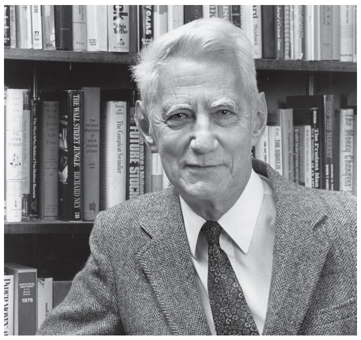
# 8 Codes



## 8.1 Code Vectors

*[I]f the machine can detect an error, why can't it locate the position of the error and correct it?*
—Richard W. Hamming
Quoted in Thomas M. Thompson
*From Error-Correcting Codes Through Sphere Packings to Simple Groups*
Carus Mathematical Monographs 21, Mathematical Association of America, 1983, p. 17


Lucent Technologies Bell Labs

The modern theory of codes originated with the work of the American mathematician and computer scientist Claude Shannon (1916–2001), whose 1937 thesis showed how algebra could play a role in the design and analysis of electrical circuits. Shannon would later be instrumental in the formation of the field of *information theory* and give the theoretical basis for what are now called error-correcting codes.

Throughout history, people have transmitted information using codes. Sometimes the intent is to disguise the message being sent, such as when each letter in a word is replaced by a different letter according to a substitution rule. Although fascinating, these secret codes, or ciphers, are not of concern here; they are the focus of the field of *cryptography*. Rather, we will concentrate on codes that are used when data must be transmitted electronically.

A familiar example of such a code is Morse code, with its system of dots and dashes. The advent of digital computers in the 20th century led to the need to transmit massive amounts of data quickly and accurately. Computers are designed to encode data as sequences of 0s and 1s. Many recent technological advancements depend on codes, and we encounter them every day without being aware of them: Satellite communications, compact disc players, the universal product codes (UPC) associated with the bar codes found on merchandise, and the international standard book numbers (ISBN) found on every book published today are but a few examples.

In this section, we will use vectors to design codes for detecting errors that may occur in the transmission of data. In later sections, we will construct codes that can not only detect but also correct errors. The vectors that arise in the study of codes are not vectors in $\mathbb{R}^n$ but vectors in $\mathbb{Z}_2^n$ or, more generally, $\mathbb{Z}_m^n$. We first encountered such vectors in Section 1.1. Since computers represent data in tems of 0s and 1s (which can be interpreted as off/on, closed/open, false/true, or no/yes), we begin by considering binary codes, which consist of vectors with entries in $\mathbb{Z}_2$.

In practice, we have a message (consisting of words, numbers, or symbols) that we wish to transmit. We begin by encoding each "word" of the message as a binary vector.

**Definition** A ***binary code*** is a set of binary vectors (of the same length) called ***code vectors.*** The process of converting a message into code vectors is called ***encoding,*** and the reverse process is called ***decoding.***

As we will see, it is highly desirable that a code have other properties as well, such as the ability to spot when an error has occurred in the transmission of a code vector and, if possible, to suggest how to correct the error.

Suppose that we have already encoded a message as a set of binary code vectors. We now want to send the binary code vectors across a *channel* (such as a radio transmitter, a telephone line, a fiber optic cable, or a CD laser). Unfortunately, the channel may be "noisy" (because of electrical interference, competing signals, or dirt and scratches). As a result, errors may be introduced: Some of the 0s may be changed to 1s, and vice versa. How can we guard against this problem?

**Example 8.1**

We wish to encode and transmit a message consisting of one of the words *up*, *down*, *left*, or *right*. We decide to use the four vectors in $\mathbb{Z}_2^2$ as our binary code, as shown in Table 8.1.

If the receiver has this table too and the encoded message is transmitted without error, decoding is trivial. However, let's suppose that a single error occurred. (By an error, we mean that one component of the code vector changed.) For example, suppose we sent the message "down" encoded as [0, 1] but an error occurred in the transmission of the first component and the 0 changed to a 1. The receiver would then see [1, 1] instead and decode the message as "right." (We will only concern ourselves with the case of single errors such as this one. In practice, it is usually assumed that the probability of multiple errors is negligibly small.) Even if the receiver knew (somehow) that a single error had occurred, he or she would not know whether the correct code vector was [0, 1] or [1, 0].

**Table 8.1**

| Message | up | down | left | right |
|---|---|---|---|---|
| Code | [0, 0] | [0, 1] | [1, 0] | [1, 1] |

But suppose we sent the message using a code that was a subset of $\mathbb{Z}_2^3$—in other words, a binary code of length 3, as shown in Table 8.2.

**Table 8.2**

| Message | up | down | left | right |
|---|---|---|---|---|
| Code | [0, 0, 0] | [0, 1, 1] | [1, 0, 1] | [1, 1, 0] |

This code can detect any single error. For example, if "down" was sent as [0, 1, 1] and an error occurred in one component, the receiver would read either [1, 1, 1] or [0, 0, 1] or [0, 1, 0], none of which is a code vector. So the receiver would know that an error had occurred (but not where) and could ask that the encoded message be retransmitted. (Why wouldn't the receiver know where the error was?)

The term *parity* comes from the Latin word *par,* meaning "equal" or "even." Two integers are said to have the same parity if they are both even or both odd.

The code in Table 8.2 is an example of an ***error-detecting code.*** Until the 1940s, this was the best that could be achieved. The advent of digital computers led to the development of codes that could *correct* as well as detect errors. We will consider these in Sections 8.2, 8.4, and 8.5.

The message to be transmitted may itself consist of binary vectors. In this case, a simple but useful error-detecting code is a ***parity check code,*** which is created by appending an extra component—called a ***check digit***—to each vector so that the parity (the total number of 1s) is even.

**Example 8.2**

If the message to be sent is the binary vector [1, 0, 0, 1, 0, 1], which has an odd number of 1s, then the check digit will be 1 (in order to make the total number of 1s in the code vector even) and the code vector will be [1, 0, 0, 1, 0, 1, 1].

Note that a single error will be detected, since it will cause the parity of the code vector to change from even to odd. For example, if an error occurred in the third component, the code vector would be received as [1, 0, 1, 1, 0, 1, 1], whose parity is odd because it has five 1s.

Let's look at this concept a bit more formally. Suppose the message is the binary vector $\mathbf{b} = [b_1, b_2, \ldots, b_n]$ in $\mathbb{Z}_2^n$. Then the parity check code vector is $\mathbf{v} = [b_1, b_2, \ldots, b_n, d]$ in $\mathbb{Z}_2^{n+1}$, where the check digit $d$ is chosen so that

$$b_1 + b_2 + \cdots + b_n + d = 0 \text{ in } \mathbb{Z}_2$$

or, equivalently, so that

$$\mathbf{1} \cdot \mathbf{v} = 0$$

where $\mathbf{1} = [1, 1, \ldots, 1]$, a vector whose every component is 1. The vector $\mathbf{1}$ is called a ***check vector.*** If vector $\mathbf{v}'$ is received and $\mathbf{1} \cdot \mathbf{v}' = 1$, then we can be certain that an error has occurred. (Although we are not considering the possibility of more than one error, observe that this scheme will not detect an even number of errors.)

Parity check codes are a special case of the more general ***check digit codes,*** which we will consider after first extending the foregoing ideas to more general settings. Codes using $m$-ary vectors are called ***m-ary codes.*** The next example is a direct extension of Example 8.2 to ternary codes.

**Example 8.3**

Let $\mathbf{b} = [b_1, b_2, \ldots, b_n]$ be a vector in $\mathbb{Z}_3^n$. Then a check digit code vector may be defined by $\mathbf{v} = [b_1, b_2, \ldots, b_n, d]$ (in $\mathbb{Z}_3^{n+1}$), with the check digit $d$ chosen so that

$$\mathbf{1} \cdot \mathbf{v} = 0$$

(where the check vector $\mathbf{1} = [1, 1, \ldots, 1]$ is the vector of 1s in $\mathbb{Z}_3^{n+1}$); that is, the check digit satisfies

$$b_1 + b_2 + \cdots + b_n + d = 0 \text{ in } \mathbb{Z}_3$$

For example, consider the vector $\mathbf{u} = [2, 2, 0, 1, 2]$. The sum of its components is $2 + 2 + 0 + 1 + 2 = 1$, so the check digit must be 2 (since $1 + 2 = 0$). Therefore, the associated code vector is $\mathbf{v} = [2, 2, 0, 1, 2, 2]$.

While simple check digit codes will detect single errors, it is often important to catch other common types of errors as well, such as the accidental interchange, or *transposition*, of two adjacent components. (For example, transposing the second and third components of **v** in Example 8.3 would result in the incorrect vector $\mathbf{v}' = [2, 0, 2, 1, 2, 2]$.) For such purposes, other types of check digit codes have been designed. Many of these simply replace the check vector **1** by some other carefully chosen vector **c**.

## Example 8.4

The Universal Product Code, or UPC (Figure 8.1), is a code associated with the bar codes found on many types of merchandise.

The black and white bars that are scanned by a laser at a store's checkout counter correspond to a 10-ary vector $\mathbf{u} = [u_1, u_2, \ldots, u_{11}, d]$ of length 12. The first 11 components form a vector in $\mathbb{Z}_{10}^{11}$ that gives manufacturer and product information; the last component $d$ is a check digit chosen so that $\mathbf{c} \cdot \mathbf{u} = 0$ in $\mathbb{Z}_{10}$, where the check vector **c** is the vector $[3, 1, 3, 1, 3, 1, 3, 1, 3, 1, 3, 1]$. That is, after rearranging,

$$3(u_1 + u_3 + u_5 + u_7 + u_9 + u_{11}) + (u_2 + u_4 + u_6 + u_8 + u_{10}) + d = 0$$

where $d$ is the check digit. In other words, the check digit is chosen so that the left-hand side of this expression is a multiple of 10.

0 74927 02094 6

**Figure 8.1**
A Universal Product Code

For the UPC shown in Figure 8.1, we can determine that the check digit is 6, performing all calculations in $\mathbb{Z}_{10}$:

$$\begin{aligned}\mathbf{c} \cdot \mathbf{u} &= 3 \cdot 0 + 7 + 3 \cdot 4 + 9 + 3 \cdot 2 + 7 + 3 \cdot 0 + 2 + 3 \cdot 0 + 9 + 3 \cdot 4 + d \\ &= 3(0 + 4 + 2 + 0 + 0 + 4) + (7 + 9 + 7 + 2 + 9) + d \\ &= 3(0) + 4 + d \\ &= 4 + d\end{aligned}$$

The check digit $d$ must be 6 to make the result of the calculation 0 in $\mathbb{Z}_{10}$.

(Another way to think of the check digit in this example is that it is chosen so that $\mathbf{c} \cdot \mathbf{u}$ will be a *multiple* of 10.)

The Universal Product Code will detect all single errors and most transposition errors in adjacent components. To see this last point, suppose that the UPC in Example 8.4 were incorrectly written as $\mathbf{u}' = [0, 7, 4, 2, 9, 7, 0, 2, 0, 9, 4, 6]$, with the fourth and fifth components transposed. When we applied the check vector, we would have $\mathbf{c} \cdot \mathbf{u}' = 4 \neq 0$ (verify this!), alerting us to the fact that there had been an error. (See Exercises 15 and 16.)

## Example 8.5

The 10-digit International Standard Book Number (ISBN-10) code is another widely used check digit code. It is designed to detect more types of errors than the Universal Product Code and, consequently, is slightly more complicated. Yet the basic principle is the same.

The code vector is a vector in $\mathbb{Z}_{11}^{10}$. The first nine components give country, publisher, and book information; the tenth component is the check digit. Suppose the ISBN-10 for a book is 0-534-34450-X. It is recorded as the vector

$$\mathbf{b} = [0, 5, 3, 4, 3, 4, 4, 5, 0, X]$$

where the check "digit" is the letter X.

For the ISBN-10 code, the check vector is the vector $\mathbf{c} = [10, 9, 8, 7, 6, 5, 4, 3, 2, 1]$, and we require that $\mathbf{c} \cdot \mathbf{b} = 0$ in $\mathbb{Z}_{11}$. Let's determine the check digit for the vector **b** in this example. We must compute

$$\mathbf{c} \cdot \mathbf{b} = 10 \cdot 0 + 9 \cdot 5 + 8 \cdot 3 + 7 \cdot 4 + 6 \cdot 3 + 5 \cdot 4 + 4 \cdot 4 + 3 \cdot 5 + 2 \cdot 0 + d$$

where $d$ is the check digit. We begin by performing all of the multiplications in $\mathbb{Z}_{11}$. (For example, $9 \cdot 5 = 1$, since 45 is 1 more than the closest smaller multiple of 11—namely, 44. On an 11-hour clock, 45 o'clock is 1 o'clock.) The simplified sum is

$$0 + 1 + 2 + 6 + 7 + 9 + 5 + 4 + 0 + d$$

and adding in $\mathbb{Z}_{11}$ leaves us with $1 + d$. The check digit $d$ must now be chosen so that the final result is 0; therefore, in $\mathbb{Z}_{11}$, $d = 10$. (Equivalently, $d$ must be chosen so that $\mathbf{c} \cdot \mathbf{b}$ will be a multiple of 11.) But since it is preferable that each component of an ISBN-10 be a single digit, the Roman numeral X is used for 10 whenever it occurs as a check digit, as it does here.

The ISBN code will detect all single errors and adjacent transposition errors (see Exercises 19–21).

**Remark:** The ISBN-10 code has been used since 1970. However, since 2007, most books have also been identified with a 13-digit ISBN code. This ISBN-13 code is compatible with the 13-digit European Article Number code (EAN-13) and uses a check digit scheme similar to the UPC. Specifically, an ISBN-13 code is a vector in $\mathbb{Z}_{10}^{13}$ where the last digit is the check digit and the check vector is $[1, 3, 1, 3, \ldots, 3, 1]$ in $\mathbb{Z}_{10}^{13}$. Like its UPC cousin, the ISBN-13 code will detect all single errors but not all adjacent transposition errors.

## Exercises 8.1

### Code Vectors

*In Exercises 1 and 2, find the parity check code vector for the binary vector* **u.**

**1.** $\mathbf{u} = [1, 0, 1, 1]$ **2.** $\mathbf{u} = [1, 1, 0, 1, 1]$

*In Exercises 3–6, a parity check code vector* **v** *is given. Determine whether a single error could have occurred in the transmission of* **v.**

**3.** $\mathbf{v} = [1, 0, 1, 0]$ **4.** $\mathbf{v} = [1, 1, 1, 0, 1, 1]$

**5.** $\mathbf{v} = [0, 1, 0, 1, 1, 1]$ **6.** $\mathbf{v} = [1, 1, 0, 1, 0, 1, 1, 1]$

*Exercises 7–10 refer to check digit codes in which the check vector is the vector* $\mathbf{c} = [1, 1, \ldots, 1]$ *of the appropriate length. In each case, find the check digit* d *that would be appended to the vector* **u.**

**7.** $\mathbf{u} = [1, 2, 2, 2]$ in $\mathbb{Z}_3^4$ **8.** $\mathbf{u} = [3, 4, 2, 3]$ in $\mathbb{Z}_5^4$

**9.** $\mathbf{u} = [1, 5, 6, 4, 5]$ in $\mathbb{Z}_7^5$ **10.** $\mathbf{u} = [3, 0, 7, 5, 6, 8]$ in $\mathbb{Z}_9^6$

**11.** Prove that for any positive integers $m$ and $n$, the check digit code in $\mathbb{Z}_m^n$ with check vector $\mathbf{c} = \mathbf{1} = [1, 1, \ldots, 1]$ will detect all single errors. (That is, prove that if vectors **u** and **v** in $\mathbb{Z}_m^n$ differ in exactly one entry, then $\mathbf{c} \cdot \mathbf{u} \neq \mathbf{c} \cdot \mathbf{v}$.)

*In Exercises 12 and 13, find the check digit* d *in the given Universal Product Code.*

**12.** $[0, 5, 9, 4, 6, 4, 7, 0, 0, 2, 7, d]$

**13.** $[0, 1, 4, 0, 1, 4, 1, 8, 4, 1, 2, d]$

**14.** Consider the UPC $[0, 4, 6, 9, 5, 6, 1, 8, 2, 0, 1, 5]$.
- **(a)** Show that this UPC cannot be correct.
- **(b)** Assuming that a single error was made and that the incorrect digit is the 6 in the third entry, find the correct UPC.

**15.** Prove that the Universal Product Code will detect all single errors.

**16.** **(a)** Prove that if a transposition error is made in the second and third entries of the UPC $[0, 7, 4, 9, 2, 7, 0, 2, 0, 9, 4, 6]$, the error will be detected.
- **(b)** Show that there is a transposition involving two adjacent entries of the UPC in part (a) that would not be detected.
- **(c)** In general, when will the Universal Product Code not detect a transposition error involving two adjacent entries?

*In Exercises 17 and 18, find the check digit* d *in the given International Standard Book Number (ISBN-10).*

**17.** [0, 3, 8, 7, 9, 7, 9, 9, 3, $d$]

**18.** [0, 3, 9, 4, 7, 5, 6, 8, 2, $d$]

**19.** Consider the ISBN-10 [0, 4, 4, 9, 5, 0, 8, 3, 5, 6].

**(a)** Show that this ISBN-10 cannot be correct.

**(b)** Assuming that a single error was made and that the incorrect digit is the 5 in the fifth entry, find the correct ISBN-10.

**20. (a)** Prove that if a transposition error is made in the fourth and fifth entries of the ISBN-10 [0, 6, 7, 9, 7, 6, 2, 9, 0, 6], the error will be detected.

**(b)** Prove that if a transposition error is made in any two adjacent entries of the ISBN-10 in part (a), the error will be detected.

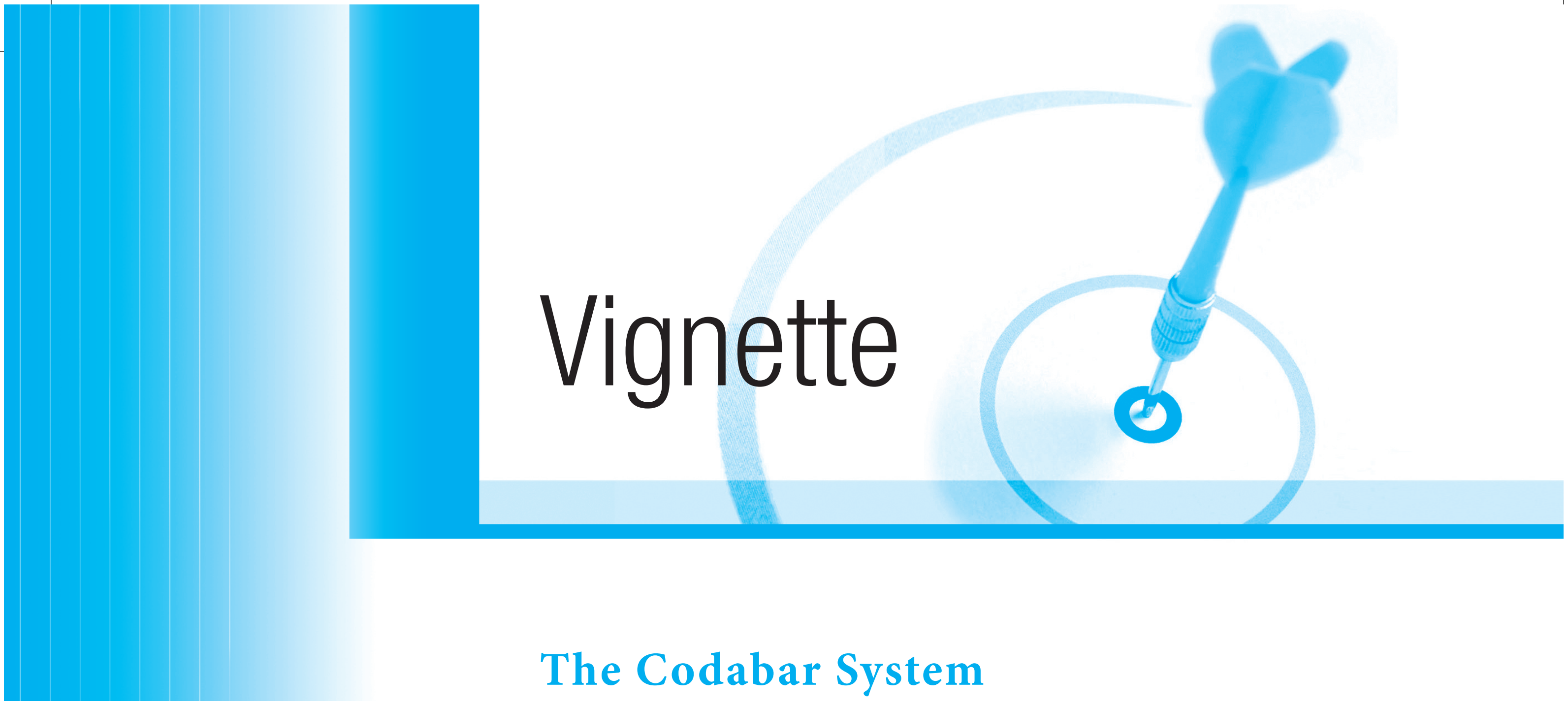
**(c)** Prove, in general, that the ISBN-10 code will always detect a transposition error involving two adjacent entries.

**21.** Consider the ISBN-10 [0, 8, 3, 7, 0, 9, 9, 0, 2, 6].

**(a)** Show that this ISBN-10 cannot be correct.

**(b)** Assuming that the error was a transposition error involving two adjacent entries, find the correct ISBN-10.

**(c)** Give an example of an ISBN-10 for which a transposition error involving two adjacent entries will be detected but will not be correctable.



## The Codabar System

Every credit card and ATM card is uniquely identified by a 16-digit number that represents a check digit code. As in the examples in this section, the first 15 digits are assigned by the bank issuing the card, whereas the last digit is a check digit determined by a formula that uses modular arithmetic.

All the major banks use a system called Codabar to assign the check digit. It is a slight variation on the method of the Universal Product Code and is based on an algorithm devised by IBM computer scientist Hans Peter Luhn in the 1950s.

Suppose that the first 15 digits of your card are

5412 3456 7890 432

and that the check digit is $d$. This corresponds to the vector

$$\mathbf{x} = [5, 4, 1, 2, 3, 4, 5, 6, 7, 8, 9, 0, 4, 3, 2, d]$$

in $Z_{10}^{16}$. The Codabar system uses the check vector $\mathbf{c} = [2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1]$, but instead of requiring that $\mathbf{c} \cdot \mathbf{x} = 0$ in $\mathbb{Z}_{10}$, an extra calculation is added to increase the error-detecting capability of the code. Let $h$ count the number of digits in *odd positions* that are *greater than 4*. In this example, these digits are 5, 5, 7, and 9, so $h = 4$.

It is now required that $\mathbf{c} \cdot \mathbf{x} + h = 0$ in $\mathbb{Z}_{10}$. Thus, in the example, we have, rearranging and working modulo 10,

$$\begin{aligned}\mathbf{c} \cdot \mathbf{x} + h &= (2 \cdot 5 + 4 + 2 \cdot 1 + 2 + 2 \cdot 3 + 4 + 2 \cdot 5 + 6 + 2 \cdot 7 + 8 + 2 \cdot 9 \\ &\qquad + 0 + 2 \cdot 4 + 3 + 2 \cdot 2 + d) + 4 \\ &= 2\,(5 + 1 + 3 + 5 + 7 + 9 + 4 + 2) + (4 + 2 + 4 + 6 + 8 + 0 \\ &\qquad + 3 + d) + 4 \\ &= 2\,(6) + 7 + d + 4 \\ &= 3 + d\end{aligned}$$

Thus, the check digit $d$ for this card must be 7, so the result of the calculation is 0 in $\mathbb{Z}_{10}$.

The Codabar system is one of the most efficient error-detecting methods. It will detect all single-digit errors and most other common errors such as adjacent transposition errors.

## 8.2 Error-Correcting Codes

Section 8.1 discussed examples of error-detecting codes. We turn now to the problem of designing codes that can *correct* as well as detect certain types of errors. Our message will be a vector $\mathbf{x}$ in $\mathbb{Z}_2^k$ for some $k$, and we will encode it by using a matrix transformation $T : \mathbb{Z}_2^k \to \mathbb{Z}_2^n$ for some $n > k$. The vector $T(\mathbf{x})$ will be called a ***code vector.*** A simple example will serve to illustrate the approach we will take, which is a generalization of the parity check vectors in Example 8.1.

### Example 8.6

Suppose the message is a single binary digit: 0 or 1. If we encode the message by simply repeating it twice, then the code vectors are [0, 0] and [1, 1]. This code can detect single errors. For example, if we transmit [0, 0] and an error occurs in the first component, then [1, 0] is received and an error is detected, because this is not a legal code vector. However, the receiver cannot correct the error, since [1, 0] would also be the result of an error in the second component if [1, 1] had been transmitted.

We can solve this problem by making the code vectors longer—repeating the message digit three times instead of two. Thus, 0 and 1 are encoded as [0, 0, 0] and [1, 1, 1], respectively. Now if a single error occurs, we cannot only detect it but also correct it. For example, if [0, 1, 0] is received, then we know it must have been the result of a single error in the transmission of [0, 0, 0], since a single error in [1, 1, 1] could not have produced it.

Note that the code in Example 8.6 can be achieved by means of a matrix transformation, albeit a particularly trivial one. Let $G = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$ and define $T : \mathbb{Z}_2 \to \mathbb{Z}_2^3$ by $T(\mathbf{x}) = G\mathbf{x}$. (Here we are thinking of the elements of $\mathbb{Z}_2$ as $1 \times 1$ matrices.) The matrix $G$ is called a ***generator matrix*** for the code.

To tell whether a received vector is a code vector, we perform not one but *two* parity checks. We require that the received vector $\mathbf{c} = \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$ satisfies $c_1 = c_2 = c_3$. We can write these equations as a linear system over $\mathbb{Z}_2$:

$$\begin{aligned} c_1 &= c_2 \\ c_1 &= c_3 \end{aligned} \quad \text{or} \quad \begin{aligned} c_1 + c_2 \phantom{+ c_3} &= 0 \\ c_1 \phantom{+ c_2} + c_3 &= 0 \end{aligned} \tag{1}$$

If we let $P = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix}$, then (1) is equivalent to $P\mathbf{c} = \mathbf{0}$. The matrix $P$ is called a ***parity check matrix*** for the code. Observe that $PG = \begin{bmatrix} 0 \\ 0 \end{bmatrix} = O$.

To see how these matrices come into play in the correction of errors, suppose we send 1 as $\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = [1 \quad 1 \quad 1]^T$, but a single error causes it to be received as

$\mathbf{c}' = [1 \quad 0 \quad 1]^T$. We compute

$$P\mathbf{c}' = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \neq \mathbf{0}$$

so we know that $\mathbf{c}'$ cannot be a code vector. Where is the error? Notice that $P\mathbf{c}'$ is the second column of the parity check matrix $P$—this tells us that the error is in the second component of $\mathbf{c}'$ (which we will prove in Theorem 8.1 below) and allows us to correct the error. (Of course, in this example we could find the error faster without using matrices, but the idea is a useful one.)

To generalize the ideas in the last example, we make the following definitions.

**Definitions** If $k < n$, then any $n \times k$ matrix of the form $G = \begin{bmatrix} I_k \\ A \end{bmatrix}$, where $A$ is an $(n-k) \times k$ matrix over $\mathbb{Z}_2$, is called a ***standard generator matrix*** for an ***(n, k) binary code*** $T : \mathbb{Z}_2^k \to \mathbb{Z}_2^n$. Any $(n-k) \times n$ matrix of the form $P = [B \quad I_{n-k}]$, where $B$ is an $(n-k) \times k$ matrix over $\mathbb{Z}_2$, is called a ***standard parity check matrix.*** The code is said to have ***length*** $n$ and ***dimension*** $k$.

Here is what we need to know: (a) When is $G$ the standard generator matrix for an *error-correcting* binary code? (b) Given $G$, how do we find an associated standard parity check matrix $P$? It turns out that the answers are quite easy, as shown by the following theorem.

**Theorem 8.1**

If $G = \begin{bmatrix} I_k \\ A \end{bmatrix}$ is a standard generator matrix and $P = [B \quad I_{n-k}]$ is a standard parity check matrix, then $P$ is the parity check matrix associated with $G$ if and only if $A = B$. The corresponding $(n, k)$ binary code is (single) error-correcting if and only if the columns of $P$ are nonzero and distinct.

Before we prove the theorem, let's consider another, less trivial example that illustrates it.

**Example 8.7**

Suppose we want to design an error-correcting code that uses three parity check equations. Since these equations give rise to the rows of $P$, we have $n - k = 3$ and $k = n - 3$. The message vectors come from $\mathbb{Z}_2^k$, so we would like $k$ (and therefore $n$) to be as large as possible in order that we may transmit as much information as possible. By Theorem 8.1, the $n$ columns of $P$ need to be nonzero and distinct, so the maximum occurs when they consist of all the $2^3 - 1 = 7$ nonzero vectors of $\mathbb{Z}_2^{n-1} = \mathbb{Z}_2^3$, One such candidate is

$$P = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

This means that

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

and thus, by Theorem 8.1, a standard generator matrix for this code is

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

As an example of how the generator matrix works, suppose we encode $\mathbf{x} = [0 \quad 1 \quad 0 \quad 1]^T$ to get the code vector

$$\mathbf{c} = G\mathbf{x} = [0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0]^T$$

If this vector is received, it is seen to be correct, since $P\mathbf{c} = \mathbf{0}$. On the other hand, if $\mathbf{c}' = [0 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1 \quad 0]^T$ is received, we compute

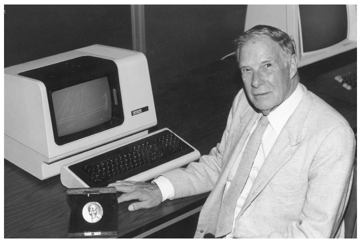
$$P\mathbf{c}' = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

which we recognize as column 3 of $P$. Therefore, the error is in the third component of $\mathbf{c}'$, and by changing it we recover the correct code vector $\mathbf{c}$. We also know that the first four components of a code vector are the original message vector, so in this case we decode $\mathbf{c}$ to get the original $\mathbf{x} = [0 \quad 1 \quad 0 \quad 1]^T$.

The code in Example 8.7 is called the (7, 4) Hamming code. Any binary code constructed in this fashion is called an ***(n, k)* Hamming code.** Observe that, by construction, an $(n, k)$ Hamming code has $n = 2^{n-k} - 1$.

**Proof of Theorem 8.1** (Throughout this proof we denote by $\mathbf{a}_i$ the $i$th column of a matrix $A$.) With $P$ and $G$ as in the statement of the theorem, assume first that they are standard parity check and generator matrices for the same binary code. Therefore, for every $\mathbf{x}$ in $\mathbb{Z}_2^k$, $PG\mathbf{x} = \mathbf{0}$. In terms of block multiplication,

$$[B \quad I]\begin{bmatrix} I \\ A \end{bmatrix}\mathbf{x} = \mathbf{0} \quad \text{for all } \mathbf{x} \text{ in } \mathbb{Z}_2^k$$


Naval Postgraduate School

Richard W. Hamming (1915–1998) received his Ph.D. in mathematics from the University of Illinois at Urbana-Champaign in 1942. His mathematical research interests were in the fields of differential equations and numerical analysis. From 1946 to 1976, he worked at Bell Labs, after which he joined the faculty at the U.S. Naval Postgraduate School in Monterey, California. In 1950, he published his fundamental paper on error-correcting codes, giving an explicit construction for the optimal codes Claude Shannon had proven theoretically possible in 1948.

Equivalently, for all $\mathbf{x}$ in $\mathbb{Z}_2^k$, we have

$$B\mathbf{x} + A\mathbf{x} = (B + A)\mathbf{x} = (BI + IA)\mathbf{x} = [B \quad I]\begin{bmatrix} I \\ A \end{bmatrix}\mathbf{x} = \mathbf{0}$$

or

$$B\mathbf{x} = A\mathbf{x}$$

If we now take $\mathbf{x} = \mathbf{e}_i$, the $i$th standard basis vector in $\mathbb{Z}_2^k$, we see that

$$\mathbf{b}_i = B\mathbf{e}_i = A\mathbf{e}_i = \mathbf{a}_i \quad \text{for all } i$$

Therefore, $B = A$.

Conversely, it is easy to check that if $B = A$, then $PG\mathbf{x} = \mathbf{0}$ for every $\mathbf{x}$ in $\mathbb{Z}_2^k$ (see Exercise 14).

To see that such a pair determines an error-correcting code if the columns of $P$ are nonzero and distinct, let $\mathbf{x}$ be a message vector in $\mathbb{Z}_2^k$ and let the corresponding code vector be $\mathbf{c} = G\mathbf{x}$. Then $P\mathbf{c} = \mathbf{0}$. Suppose there has been an error in the $i$th component, resulting in the vector $\mathbf{c}'$. It follows that $\mathbf{c}' = \mathbf{c} + \mathbf{e}_i$. We now compute

$$P\mathbf{c}' = P(\mathbf{c} + \mathbf{e}_i) = P\mathbf{c} + P\mathbf{e}_i = \mathbf{0} + \mathbf{p}_i = \mathbf{p}_i$$

which pinpoints the error in the $i$th component.

On the other hand, if $\mathbf{p}_i = \mathbf{0}$, then an error in the $i$th component will not be detected (i.e., $P\mathbf{c}' = \mathbf{0}$), and if $\mathbf{p}_i = \mathbf{p}_j$, then we cannot determine whether an error occurred in the $i$th or the $j$th component (Exercise 15).

The main ideas of this section are summarized below.

1. For $n > k$, an $n \times k$ matrix $G$ and an $(n - k) \times n$ matrix $P$ (with entries in $\mathbb{Z}_2$) are a standard generator matrix and a standard parity check matrix, respectively, for an $(n, k)$ binary code if and only if, in block form,

$$G = \begin{bmatrix} I_k \\ A \end{bmatrix} \quad \text{and} \quad P = [A \quad I_{n-k}]$$

for some $(n - \text{k}) \times k$ matrix $A$ over $\mathbb{Z}_2$.
2. $G$ encodes a message vector $\mathbf{x}$ in $\mathbb{Z}_2^k$ as a code vector $\mathbf{c}$ in $\mathbb{Z}_2^n$ via $\mathbf{c} = G\mathbf{x}$.
3. $G$ is error-correcting if and only if the columns of $P$ are nonzero and distinct. A vector $\mathbf{c}'$ in $\mathbb{Z}_2^n$ is a code vector if and only if $P\mathbf{c}' = \mathbf{0}$. In this case, the corresponding message vector is the vector $\mathbf{x}$ in $\mathbb{Z}_2^k$ consisting of the first $k$ components of $\mathbf{c}'$. If $P\mathbf{c}' \neq \mathbf{0}$, then $\mathbf{c}'$ is not a code vector and $P\mathbf{c}'$ is one of the columns of $P$. If $P\mathbf{c}'$ is the $i$th column of $P$, then the error is in the $i$th component of $\mathbf{c}'$ and we can recover the correct code vector (and hence the message) by changing this component.

## Exercises 8.2

### Error-Correcting Codes

**1.** Suppose we encode the four vectors in $\mathbb{Z}_2^2$ by repeating the vector twice. Thus, we have

$$\begin{aligned}[0, 0] &\to [0, 0, 0, 0]\\ [0, 1] &\to [0, 1, 0, 1]\\ [1, 0] &\to [1, 0, 1, 0]\\ [1, 1] &\to [1, 1, 1, 1]\end{aligned}$$

Show that this code is not error-correcting.

**2.** Suppose we encode the binary digits 0 and 1 by repeating each digit five times. Thus,

$$\begin{aligned}0 &\to [0, 0, 0, 0, 0]\\ 1 &\to [1, 1, 1, 1, 1]\end{aligned}$$

Show that this code can correct double errors.

*What is the result of encoding the messages in Exercises 3–5 using the (7, 4) Hamming code of Example 8.7?*

**3.** $\mathbf{x} = \begin{bmatrix}1\\1\\0\\0\end{bmatrix}$ **4.** $\mathbf{x} = \begin{bmatrix}0\\1\\1\\1\end{bmatrix}$ **5.** $\mathbf{x} = \begin{bmatrix}1\\1\\1\\1\end{bmatrix}$

*When the (7, 4) Hamming code of Example 8.7 is used, suppose the messages* $\mathbf{c}'$ *in Exercises 6–8 are received. Apply the standard parity check matrix to* $\mathbf{c}'$ *to determine whether an error has occurred and correctly decode* $\mathbf{c}'$ *to recover the original message vector* $\mathbf{x}$.

**6.** $\mathbf{c}' = [0 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1]^T$

**7.** $\mathbf{c}' = [1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 1 \quad 0]^T$

**8.** $\mathbf{c}' = [0 \quad 0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0]^T$

**9.** The parity check code in Example 8.1 is a code $\mathbb{Z}_2^6 \to \mathbb{Z}_2^7$.

**(a)** Find a standard parity check matrix for this code.

**(b)** Find a standard generator matrix.

**(c)** Apply Theorem 8.1 to explain why this code is not error-correcting.

**10.** Define a code $\mathbb{Z}_2^2 \to \mathbb{Z}_2^5$ using the standard generator matrix

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

**(a)** List all four code words.

**(b)** Find the associated standard parity check matrix for this code. Is this code (single) error-correcting?

**11.** Define a code $\mathbb{Z}_2^3 \to \mathbb{Z}_2^6$ using the standard generator matrix

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

**(a)** List all eight code words.

**(b)** Find the associated standard parity check matrix for this code. Is this code (single) error-correcting?

**12.** Show that the code in Example 8.6 is a (3, 1) Hamming code.

**13.** Construct standard parity check and generator matrices for a (15, 11) Hamming code.

**14.** In Theorem 8.1, prove that if $B = A$, then $PG\mathbf{x} = \mathbf{0}$ for every $\mathbf{x}$ in $\mathbb{Z}_2^k$.

**15.** In Theorem 8.1, prove that if $\mathbf{p}_i = \mathbf{p}_j$, then we cannot determine whether an error occurs in the $i$th or the $j$th component of the received vector.

## 8.3 Dual Codes

There are many ways of constructing new codes from old ones. In this section, we consider one of the most important of these.

First, we need to generalize the concepts of a generator and a parity check matrix for a code. Recall from Section 8.2 that a standard generator matrix for a code is an $n \times m$ matrix of the form

$$G = \left[\begin{array}{c} I_m \\ \hdashline A \end{array}\right]$$

and a standard parity check matrix is an $(n - m) \times n$ matrix of the form

$$P = [B \,\vdots\, I_{n-m}]$$

Observe that the form of these matrices guarantees that the columns of $G$ are linearly independent and the rows of $P$ are linearly independent. (Why?) In proving Theorem 8.1, we showed that $G$ and $P$ are associated with the *same* code if and only if $A = B$, which is equivalent to requiring that $PG = O$. We use these properties as the basis for the following definition.

**Definition** For $n > m$, an $n \times m$ matrix $G$ and an $(n - k) \times n$ matrix $P$ (with entries in $\mathbb{Z}_2$) are a ***generator matrix*** and a ***parity check matrix,*** respectively, for an $(n, k)$ binary code $C$ if the following conditions are all satisfied:

1. The columns of $G$ are linearly independent.
2. The rows of $P$ are linearly independent.
3. $PG = O$

Notice that property (3) implies that every column $\mathbf{v}$ of $G$ satisfies $P\mathbf{v} = \mathbf{0}$ and so is a code vector in $C$. Also, a vector $\mathbf{y}$ is in $C$ if and only if it is obtained from the generator matrix as $\mathbf{y} = G\mathbf{u}$ for some vector $\mathbf{u}$ in $\mathbb{Z}_2^n$. In other words, $C$ is the column space of $G$.

To understand the relationship between different generator matrices for the same code, we only need to recall that, just as elementary row operations do not affect the row space of a matrix (by Theorem 3.20), elementary column operations do not affect the column space. For a matrix over $\mathbb{Z}_2$, there are only two relevant operations: interchange two columns (C1) and add one column to another column (C2). (Why are these the only two elementary column operations on matrices over $\mathbb{Z}_2$?)

Similarly, elementary row operations preserve the linear independence of the rows of $P$. Moreover, if $E$ is an elementary matrix and $\mathbf{c}$ is a code vector, then

$$(EP)\mathbf{c} = E(P\mathbf{c}) = E\mathbf{0} = \mathbf{0}$$

It follows that $EP$ is also a parity check matrix for $C$. Thus, any parity check matrix can be converted into another one by means of a sequence of row operations: interchange two rows (R1) and add one row to another row (R2). We are interested in showing that any generator or parity check matrix can be brought into standard form. There is one other definition we need. We will call two codes $C_1$ and $C_2$ ***equivalent*** if there is a permutation matrix $M$ such that

$$\{M\mathbf{x} : \mathbf{x} \text{ in } C_1\} = C_2$$

In other words, if we permute the entries of the vectors in $C_1$ (all in the same way), we can obtain $C_2$. For example,

$$C_1 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \right\} \quad \text{and} \quad C_2 = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \right\}$$

are equivalent via the permutation matrix $M = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$. Permuting the entries of code vectors corresponds to permuting the *rows* of a generator matrix and permuting the *columns* of a parity check matrix for the code. (Why?)

We can bring any generator matrix for a code into standard form by means of operations C1, C2, and R1. If R1 has not been used, then we have the same code; if R1 has been used, then we have an equivalent code. We can bring any parity check matrix for a code into standard form by means of operations R1, R2, and C1. If C1 has not been used, then we have the same code; if C1 has been used, then we have an equivalent code.

The following examples illustrate these points.

**Example 8.8**

(a) Bring the generator matrix

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

into standard form and find an associated parity check matrix.

(b) Bring the parity check matrix

$$P = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

into standard form and find an associated generator matrix.

**Solution** (a) We can bring the generator matrix $G$ into standard form as follows:

$$G = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \xrightarrow{R_2 \leftrightarrow R_3} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{bmatrix} = \begin{bmatrix} I \\ \hdashline A \end{bmatrix} = G'$$

(Do you see why it is not possible to obtain standard form without using R1?) Hence, $A = [1 \quad 0]$, so

$$P = [A \,\vdots\, I] = [1 \quad 0 \quad 1]$$

is an associated parity check matrix, by Theorem 8.1.

(b) We use elementary row operations to bring $P$ into standard form, keeping in mind that we want to create an identity matrix on the *right*—not on the left, as in Gauss-Jordan elimination. We compute

$$P = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{R_1 \leftrightarrow R_2} \begin{bmatrix} 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix} \xrightarrow{R_1 + R_2} \begin{bmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} = [A \,\vdots\, I] = P'$$

Thus, $A = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$, so, by Theorem 8.1, an associated generator matrix is

$$G = \begin{bmatrix} I \\ \hdashline A \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$$

**Remark** In part (a), it is instructive to verify that $G$ and $G'$ generate equivalent, but not identical, codes. Check that this is so by computing $\{G\mathbf{x} : \mathbf{x} \text{ in } \mathbb{Z}_2^2\}$ and $\{G'\mathbf{x} : \mathbf{x} \text{ in } \mathbb{Z}_2^2\}$.

We now turn our attention to the main topic of this section, the notion of a dual code.

**Definition** Let $C$ be a set of code vectors in $\mathbb{Z}_2^n$. The orthogonal complement of $C$ is called the ***dual code*** of $C$ and is denoted $C^\perp$. That is,

$$C^\perp = \{\mathbf{x} \text{ in } \mathbb{Z}_2^n : \mathbf{c}\cdot\mathbf{x} = 0 \quad \text{for all } \mathbf{c} \text{ in } C\}$$

The dot product in $\mathbb{Z}_2^n$ behaves like the dot product in $\mathbb{R}^n$, with one important exception: Property (d) of Theorem 1.2 is no longer true. In other words, in $\mathbb{Z}_2^n$, a nonzero vector can be orthogonal to itself! As an example, take $\mathbf{x} = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$ in $\mathbb{Z}_2^2$. Then $\mathbf{x}\cdot\mathbf{x} = 1 + 1 = 0$.

**Example 8.9** Find the dual code of the code $C$ in Example 8.8(b).

**Solution** The code $C$ is

$$C = \{G\mathbf{x} : \mathbf{x} \text{ in } \mathbb{Z}_2^2\}$$

$$= \left\{ G\begin{bmatrix} 0 \\ 0 \end{bmatrix}, G\begin{bmatrix} 0 \\ 1 \end{bmatrix}, G\begin{bmatrix} 1 \\ 0 \end{bmatrix}, G\begin{bmatrix} 1 \\ 1 \end{bmatrix} \right\} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right\}$$

[Alternatively, $C = \{\mathbf{c} \text{ in } \mathbb{Z}^4 : P\mathbf{c} = \mathbf{0}\} = \text{null}(P)$. Check that this really does give the same code.]

To find $C^\perp$, we need those vectors in $\mathbb{Z}_2^4$ that are orthogonal to all four vectors in $C$. Since there are only 16 vectors altogether in $\mathbb{Z}_2^4$, we could proceed by trial and error—but here is a better method. Let $\mathbf{y} = [y_1 \quad y_2 \quad y_3 \quad y_4]^T$ be in $C^\perp$. Since $\mathbf{y}\cdot\mathbf{c} = 0$ for each $\mathbf{c}$ in $C$, we have four equations, one of which we can ignore, since it just says $0 = 0$. The other three are

$$\begin{aligned} y_2 + y_3 \phantom{{}+y_4} &= 0 \\ y_1 + \phantom{y_2 +{}} y_3 + y_4 &= 0 \\ y_1 + y_2 + \phantom{y_3 +{}} y_4 &= 0 \end{aligned}$$

Solving this system, we obtain

$$\left[\begin{array}{cccc|c} 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \end{array}\right] \longrightarrow \left[\begin{array}{cccc|c} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{array}\right]$$

(Check this.) It follows that $y_1 = y_3 + y_4$ and $y_2 = y_3$, so

$$C^{\perp} = \left\{ \begin{bmatrix} y_3 + y_4 \\ y_3 \\ y_3 \\ y_4 \end{bmatrix} \right\} = \left\{ y_3 \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} + y_4 \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

We now examine the relationship between the generator and parity check matrices of a code and its dual.

**Theorem 8.2** If $C$ is an $(n, k)$ binary code with generator matrix $G$ and parity check matrix $P$, then $C^{\perp}$ is an $(n, n - k)$ binary code such that

a. $G^T$ is a parity check matrix for $C^{\perp}$.
b. $P^T$ is a generator matrix for $C^{\perp}$.

**Proof** By definition, $G$ is an $n \times k$ matrix with linearly independent columns, $P$ is an $(n - k) \times n$ matrix with linearly independent rows, and $PG = O$. Therefore, the rows of $G^T$ are linearly independent, the columns of $P^T$ are linearly independent, and

$$G^T P^T = (PG)^T = O^T = O$$

This shows that $G^T$ is a parity check matrix for $C^{\perp}$ and $P^T$ is a generator matrix for $C^{\perp}$. Since $P^T$ is $n \times (n - k)$, $C^{\perp}$ is an $(n, n - k)$ code.

**Example 8.10** Find generator and parity check matrices for the dual code $C^{\perp}$ from Example 8.9.

**Solution** There are two ways to proceed. We will illustrate both approaches.

Method 1: According to Theorem 8.2(b), a generator matrix $G^{\perp}$ for $C^{\perp}$ is given by

$$G^{\perp} = P^T = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

This matrix is in standard form with $A = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$, so a parity check matrix for $C^{\perp}$ is

$$P^{\perp} = [A \;\; I] = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$$

by Theorem 8.1.

Method 2: Using Theorem 8.2(a) and referring to Example 8.8(b), we obtain a parity check matrix $P^\perp$ for $C^\perp$ as follows:

$$P^\perp = G^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

This matrix is not in standard form, so we use elementary row operations to convert it to

$$P^\perp = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \longrightarrow \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} = [A \;\; I] = P_1^\perp$$

Now we can use Theorem 8.1 to obtain a generator matrix $G^\perp$ for $C^\perp$:

$$G^\perp = \begin{bmatrix} I \\ A \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 0 & 1 \\ 1 & 1 \end{bmatrix}$$

**Example 8.11**

Let $C$ be the code with generator matrix

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

List the vectors in $C$ and $C^\perp$.

**Solution** The code $C$ is

$$C = \{G\mathbf{x} : \mathbf{x} \text{ in } \mathbb{Z}_2^4\} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

(Note that $C$ is a double repetition code that encodes vectors from $\mathbb{Z}_2^2$ as vectors in $\mathbb{Z}_2^4$ by writing the entries twice. See Exercise 1 in Section 8.2.) Using Theorem 8.2, we find the parity check matrix $P^\perp$ for $C^\perp$ to be

$$P^\perp = G^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}^T = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

Thus, $P^\perp$ has the form $[A \quad I]$, where $A = I$, so a generator matrix $G^\perp$ for $C^\perp$ is

$$G^\perp = \begin{bmatrix} I \\ A \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = G$$

Hence, $C^\perp$ has the same generator matrix as $C$, so $C^\perp = C$!

Courtesy Walter MacWilliams

F. Jessie MacWilliams (1917–1990) was one of the pioneers of coding theory. She received her B.A. and M.A. from Cambridge University in 1938–39, following which she studied in the United States at Johns Hopkins University and Harvard University. After marrying and raising a family, MacWilliams took a job as a computer programmer at Bell Laboratories in Murray Hill, New Jersey, in 1958, where she became interested in coding theory. In 1961, she returned to Harvard for a year and obtained a Ph.D.

Her thesis contains one of the most powerful theorems in coding theory. Now known as the **MacWilliams Identities,** this theorem relates the weight distribution (the number of codewords of each possible weight) of a linear code to the weight distribution of its dual code. The MacWilliams Identities are widely used by coding theorists, both to obtain new theoretical information about error-correcting codes and to determine the weight distributions of specific codes.

MacWilliams is perhaps best known for her book *The Theory of Error-Correcting Codes* (1977), written with N. J. A. Sloane of Bell Labs. This book is often referred to as the "bible of coding theory." In 1980, MacWilliams gave the inaugural Emmy Noether Lecture of the Association for Women in Mathematics.

A code $C$ with the property that $C^\perp = C$ is called ***self-dual.*** We can check that the code in Example 8.11 is self-dual by showing that every vector in $C$ is orthogonal to all the vectors in $C$, including itself. (Do this.)

You may have noticed that in the self-dual code in Example 8.11, every vector in $C$ has an *even* number of 1s. We will prove that this is true for every self-dual code. The following definition is useful.

**Definition** Let $\mathbf{x}$ be a vector in $\mathbb{Z}_2^n$. The ***weight*** of $\mathbf{x}$, denoted $w(\mathbf{x})$, is the number of 1s in $\mathbf{x}$.

For example, $w([1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 1]^T) = 4$. If we temporarily think of $\mathbf{x}$ as a vector in $\mathbb{R}^n$, then we can give the following alternative descriptions of $w(\mathbf{x})$. Let $\mathbf{1}$ denote the vector (of the same length as $\mathbf{x}$) all of whose entries are 1. Then $w(\mathbf{x}) = \mathbf{x} \cdot \mathbf{1}$ and $w(\mathbf{x}) = \mathbf{x} \cdot \mathbf{x}$. We can now prove the following interesting facts about self-dual codes.

**Theorem 8.3** If $C$ is a self-dual code, then:

a. Every vector in $C$ has even weight.
b. $\mathbf{1}$ is in $C$.

**Proof** (a) A vector $\mathbf{x}$ in $\mathbb{Z}_2^n$ has even weight if and only if $w(\mathbf{x}) = 0$ in $\mathbb{Z}_2$. But

$$w(\mathbf{x}) = \mathbf{x} \cdot \mathbf{x} = 0$$

since $C$ is self-dual.

(b) Using property (a), we have $\mathbf{1} \cdot \mathbf{x} = w(\mathbf{x}) = 0$ in $\mathbb{Z}_2$ for all $\mathbf{x}$ in $C$. This means that $\mathbf{1}$ is orthogonal to every vector in $C$, so $\mathbf{1}$ is in $C^\perp = C$, as required.

## Exercises 8.3

### Dual Codes

*In Exercises 1–4, G is a generator matrix for a code C. Bring G into standard form and determine whether the corresponding code is equal to C.*

**1.** $G = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \end{bmatrix}$ **2.** $G = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

**3.** $G = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ **4.** $G = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 0 \\ 1 & 0 \end{bmatrix}$

*In Exercises 5–8, P is a parity check matrix for a code C. Bring P into standard form and determine whether the corresponding code is equal to C.*

**5.** $P = \begin{bmatrix} 1 & 1 & 0 \end{bmatrix}$ **6.** $P = \begin{bmatrix} 1 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix}$

**7.** $P = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$

**8.** $P = \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$

*In Exercises 9–12, find the dual code $C^\perp$ of the code C.*

**9.** $C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \right\}$

**10.** $C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$

**11.** $C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \right\}$

**12.** $C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} \right\}$

*In Exercises 13–16, either a generator matrix G or a parity check matrix P is given for a code C. Find a generator matrix $G^\perp$ and a parity check matrix $P^\perp$ for the dual code of C.*

**13.** $G = \begin{bmatrix} 1 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$ **14.** $G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix}$

**15.** $P = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$ **16.** $P = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{bmatrix}$

**17.** Find generator and parity check matrices for the dual of the (7, 4) Hamming code in Example 8.7.

*The **even parity code** $E_n$ is the subset of $\mathbb{Z}_2^n$ consisting of all vectors with even weight. The **n-times repetition code** $\mathrm{Rep}_n$ is the subset of $\mathbb{Z}_2^n$ consisting of just the two vectors **0** and **1** (all zeros and all 1s, respectively).*

**18. (a)** Find generator and parity check matrices for $E_3$ and $\mathrm{Rep}_3$.

**(b)** Show that $E_3$ and $\mathrm{Rep}_3$ are dual to each other.

**19.** Show that $E_n$ and $\mathrm{Rep}_n$ are dual to each other.

**20.** If $C$ and $D$ are codes and $C \subseteq D$, show that $D^\perp \subseteq C^\perp$.

**21.** Show that if $C$ is a code with a generator matrix, then $(C^\perp)^\perp = C$.

**22.** Find a self-dual code of length 6.

## 8.4 Linear Codes

We now turn our attention to the most important, and most widely used, class of codes: *linear codes*. In fact, many of the examples we have already looked at fall into this category. NASA has made extensive use of linear codes to transmit pictures from outer space. In 1972, the *Mariner 9* spacecraft used a type of linear code called a *Reed-Muller code* to transmit black-and-white images of Mars (Figure 8.2). Then, between 1979 and 1981, *Voyager 1* and *Voyager 2* were able to send back remarkable color pictures of Jupiter and Saturn (reproduced in black and white in Figure 8.3) using a *Golay code,* another linear code.

**Definition** A $p$-ary ***linear code*** is a subspace $C$ of $\mathbb{Z}_p^n$.

As usual, our main interest is the case $p = 2$, the ***binary linear codes.*** Checking to see whether a subset $C$ of $\mathbb{Z}_2^n$ is a subspace involves showing that $C$ satisfies the conditions of Theorem 6.2. Since in $\mathbb{Z}_2^n$ the only scalars are 0 and 1, checking to see whether $C$ is closed under scalar multiplication only involves showing that $C$ contains the zero vector. All that remains is to check that $C$ is closed under addition.

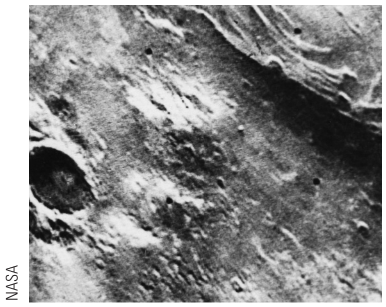


**Figure 8.2**
The southern polar cap of Mars

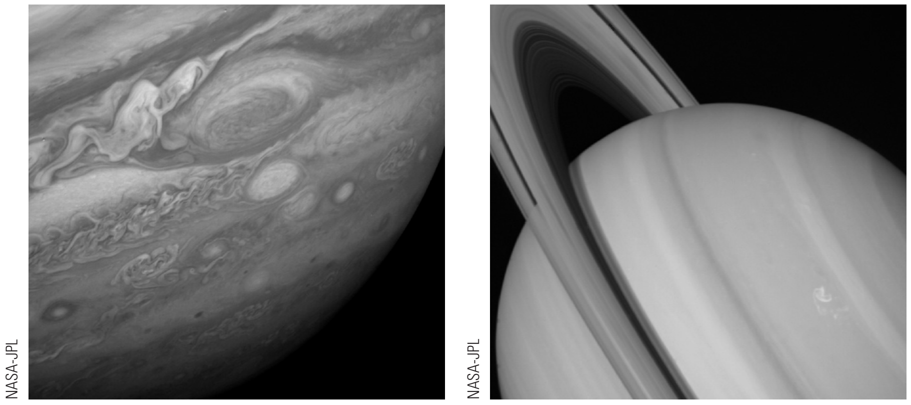


**Figure 8.3**
Jupiter's red spot and the rings of Saturn

**Example 8.12**

Are $C_1 = \left\{ \begin{bmatrix} 0\\0\\0\\0 \end{bmatrix}, \begin{bmatrix} 0\\0\\1\\1 \end{bmatrix}, \begin{bmatrix} 1\\1\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\1\\1\\1 \end{bmatrix} \right\}$ and $C_2 = \left\{ \begin{bmatrix} 0\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\0\\0 \end{bmatrix}, \begin{bmatrix} 1\\0\\1 \end{bmatrix} \right\}$ (binary)

linear codes?

**Solution** $C_1$ clearly contains the zero vector and is closed under addition, so it is a linear code. $C_2$ is not closed under addition, since it does not contain

$$\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}1\\0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\1\end{bmatrix}$$

Hence, $C_2$ is not linear.

For the remainder of this section, we will dispense with the adjective "binary," since all of the codes we will be considering will be binary codes. If a linear code $C$ is a $k$-dimensional subspace of $\mathbb{Z}_2^n$, then we say that $C$ is an ***(n, k) code.***

**Example 8.13**

(a) The code $C_1$ in Example 8.12 is a subspace of $\mathbb{Z}_2^4$ and has dimension 2, since

$$\left\{\begin{bmatrix}0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\0\end{bmatrix}\right\}$$

is a basis for $C_1$. (In fact, $C_1$ has exactly three different two-element bases. What are the other two? See Exercise 9.) Hence, $C_1$ is a (4, 2) code.

(b) The (7, 4) Hamming code $H$ introduced in Section 8.2 is a (7, 4) linear code (fortunately!), in our new terminology. It is linear because it has a generator matrix $G$, so its vectors are all the vectors of the form $G\mathbf{x}$, where $\mathbf{x}$ is in $\mathbb{Z}_2^4$. But this is just the column space of the $7 \times 4$ matrix $G$ and so is a subspace of $\mathbb{Z}_2^7$. Since the four columns of $G$ are linearly independent (why?), they form a basis for $H$. Therefore, $H$ is a (7, 4) code.

(c) The codes

$$C = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\1\end{bmatrix}\right\} \quad \text{and} \quad C^\perp = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}\right\}$$

are dual codes. It is easy to see that each of these is a linear code, that $\dim C = 1$, and that $\dim C^\perp = 2$. (Check these claims.) Therefore, $C$ is a (3, 1) code and $C^\perp$ is a (3, 2) code. The fact that $3 = 1 + 2$ is not an accident, as the next theorem shows.

**Theorem 8.4** Let $C$ be an $(n, k)$ linear code.

a. The dual code $C^\perp$ is an $(n, n - k)$ linear code.
b. $C$ contains $2^k$ vectors, and $C^\perp$ contains $2^{n-k}$ vectors.

**Proof** (a) Since $C$ is an $(n, k)$ linear code, it is a $k$-dimensional subspace of $\mathbb{Z}_2^n$. Its dual $C^\perp$ is the orthogonal complement of $C$ and so is also a subspace of $\mathbb{Z}_2^n$, by Theorem 5.9(a). Thus, $C^\perp$ is a linear code.

Now we can apply Theorem 5.13 to show that

$$\dim C^{\perp} = n - \dim C = n - k$$

(*Note:* Theorems 5.9 and 5.13 are true if $\mathbb{R}^n$ is replaced by $\mathbb{Z}_2^n$. This is the case for most of the nongeometric results about orthogonality.) It follows that $C^{\perp}$ is an $(n, n-k)$ code.

(b) Let $\{\mathbf{v}_1, \dots, \mathbf{v}_k\}$ be a basis for $C$. Then the vectors in $C$ are all the vectors of the form

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

where each $c_i$ is either 0 or 1. Therefore, there are two possibilities for $c_1$ and, for each of these, two possibilities for $c_2$, and so on, making the total number of possibilities for $\mathbf{v}$

$$\underbrace{2 \times 2 \times \cdots \times 2}_{k \text{ times}} = 2^k$$

Thus, $C$ contains exactly $2^k$ vectors. Applying this formula to its $(n, n-k)$ dual code, we see that $C^{\perp}$ has $2^{n-k}$ vectors.

The Reed-Muller codes are named after the computer scientists Irving S. Reed and David E. Muller, who published papers, independently, about these codes in 1954.

Recall that the binary, or base two, representation of a number arises from writing it as a sum of distinct powers of two. If $n = b_k \cdot 2^k + \cdots + b_1 \cdot 2 + b_0$, where each $b_i$ is 0 or 1, then in base two $n$ is represented as $n = b_i \cdots b_1 b_0$. For example, $25 = 16 + 8 + 1 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2 + 1$, so the binary representation of 25 is 11001.

We now construct one of the oldest families of linear codes, the Reed-Muller codes. As mentioned earlier, this is the type of code that was used by the *Mariner 9* spacecraft to transmit pictures of Mars. In order to be transmitted, each photograph had to be broken down into picture elements, or *pixels*. This was done by overlaying the photograph with a $700 \times 832$ pixel grid and then assigning to each pixel one of 64 shades of gray, ranging from white (0) to black (63). Since $64 = 2^6$, we can use binary arithmetic to represent each of these shades: white is 000000 and black is 111111. We can then rewrite these 64 binary numbers as vectors in $\mathbb{Z}_2^6$ and encode them using a code that corrects as many errors as possible. The code that was chosen for use by *Mariner 9* belongs to a large family of codes that are most easily defined inductively.

**Definition** The (first-order) ***Reed-Muller codes*** $R_n$ are defined inductively as follows:

1. For $n = 0$, $R_0 = \mathbb{Z}_2 = \{0, 1\}$.
2. For $n \geq 1$, $R_n$ is the subspace of $\mathbb{Z}_2^{2^n}$ whose basis consists of all vectors of the form

$$\begin{bmatrix} \mathbf{u} \\ \mathbf{u} \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix}$$

where $\mathbf{u}$ is a basis vector in $R_{n-1}$, $\mathbf{0}$ is the zero vector in $\mathbb{Z}_2^{2^{n-1}}$, and $\mathbf{1}$ is the vector of 1s in $\mathbb{Z}_2^{2^{n-1}}$.

To get a sense of what vectors these codes contain, let's use the definition to construct $R_1$ and $R_2$. A basis for $R_0 = \mathbb{Z}_2$ is just $\{1\}$, so a basis for $R_1$ is

$$\left\{ \begin{bmatrix} 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right\}$$

Thus, by closure under addition, $R_1$ must also contain the vectors

$$\begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}0\\1\end{bmatrix}=\begin{bmatrix}1\\0\end{bmatrix} \quad\text{and}\quad \begin{bmatrix}1\\1\end{bmatrix}+\begin{bmatrix}1\\1\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}$$

It is easy to check that no other vectors can be obtained by addition, so

$$R_1=\left\{\begin{bmatrix}0\\0\end{bmatrix},\begin{bmatrix}0\\1\end{bmatrix},\begin{bmatrix}1\\0\end{bmatrix},\begin{bmatrix}1\\1\end{bmatrix}\right\}=\mathbb{Z}_2^2$$

Similarly, a basis for $R_2$ is

$$\left\{\begin{bmatrix}1\\1\\1\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\\1\end{bmatrix},\begin{bmatrix}0\\0\\1\\1\end{bmatrix}\right\}$$

and, by closure under addition, it is easy to check that the $8 = 2^3$ vectors in $R_2$ are

$$R_2=\left\{\begin{bmatrix}0\\0\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\\1\end{bmatrix},\begin{bmatrix}0\\1\\0\\1\end{bmatrix},\begin{bmatrix}0\\1\\1\\0\end{bmatrix},\begin{bmatrix}1\\0\\1\\0\end{bmatrix},\begin{bmatrix}1\\0\\0\\1\end{bmatrix},\begin{bmatrix}1\\1\\0\\0\end{bmatrix},\begin{bmatrix}1\\1\\1\\1\end{bmatrix}\right\}$$

Notice that in $R_1$ every code vector except $\mathbf{0}$ and $\mathbf{1}$ has weight 1, and in $R_2$ every code vector except $\mathbf{0}$ and $\mathbf{1}$ has weight 2. This is a general property of the Reed-Muller codes, and we prove it as part of the next theorem. But first, note that the ***complement*** of a vector $\mathbf{x}$ in $\mathbb{Z}_2^n$ is the vector $\bar{\mathbf{x}}$ obtained by changing all the zeros to 1s and vice versa. For example,

$$\mathbf{x}=\begin{bmatrix}1\\1\\0\\1\end{bmatrix}\Leftrightarrow\bar{\mathbf{x}}=\begin{bmatrix}0\\0\\1\\0\end{bmatrix}$$

Observe that $\bar{\mathbf{x}} = \mathbf{x} + 1$, where $\mathbf{1}$ is the vector consisting entirely of 1s.

**Theorem 8.5** For $n \geq 1$, the Reed-Muller code $R_n$ is a $(2^n, n + 1)$ linear code in which every code vector except $\mathbf{0}$ and $\mathbf{1}$ has weight $2^{n-1}$.

**Proof** We will prove this theorem by induction on $n$. For $n = 1$, we have already seen that $R_1 = \mathbb{Z}_2^2$ is a $(2, 2) = (2^1, 1 + 1)$ linear code in which every code vector except $\mathbf{0}$ and $\mathbf{1}$ has weight $1 = 2^{1-1}$. Assume that the result is true for $n = k$; that is, assume that $R_k$ is a $(2^k, k + 1)$ linear code in which every code vector except $\mathbf{0}$ and $\mathbf{1}$ has weight $2^{k-1}$. Now consider $R_{k+1}$.

By construction, $R_{k+1}$ has a basis consisting of vectors of the form $\begin{bmatrix}\mathbf{u}\\\mathbf{u}\end{bmatrix}$, where $\mathbf{u}$ is in $R_k$, together with the vector $\begin{bmatrix}\mathbf{0}\\\mathbf{1}\end{bmatrix}$. By the induction hypothesis, the vectors $\mathbf{u}$, $\mathbf{0}$, and $\mathbf{1}$ are in $\mathbb{Z}_2^{2^k}$; hence, the basis vectors for $R_{k+1}$ are in $\mathbb{Z}_2^{2^{k+1}}$. Moreover, the dimension of

$R_k$ is $k + 1$, so there are $k + 1$ vectors of the form $\begin{bmatrix}\mathbf{u}\\ \mathbf{u}\end{bmatrix}$ and one more, $\begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}$. It follows that the dimension of $R_{k+1}$ is $k + 2$, and therefore $R_{k+1}$ is a $(2^{k+1}, k + 2)$ linear code.

For the final assertion, note that the vectors in $R_{k+1}$ are obtained as linear combinations of the basis vectors and so are of the form

$$\mathbf{v} = c_1\begin{bmatrix}\mathbf{u}_1\\ \mathbf{u}_1\end{bmatrix} + \cdots + c_{k+1}\begin{bmatrix}\mathbf{u}_{k+1}\\ \mathbf{u}_{k+1}\end{bmatrix} + c_{k+2}\begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix}$$

where $\{\mathbf{u}_1, \ldots, \mathbf{u}_{k+1}\}$ is a basis for $R_k$, $\mathbf{0}$ and $\mathbf{1}$ are in $\mathbb{Z}_2^{2^k}$, and each $c_i$ is 0 or 1. Suppose $\mathbf{v} \neq \mathbf{0}, \mathbf{1}$ and let $\mathbf{u} = c_1\mathbf{u}_1 + \cdots + c_{k+1}\mathbf{u}_{k+1}$. (Hence, $\mathbf{u}$ is in $R_k$.) If $c_{k+2} = 0$, then $\mathbf{u} \neq \mathbf{0}, \mathbf{1}$, so, by the induction hypothesis, $\mathbf{u}$ has weight $2^{k-1}$. But then $\mathbf{v}$ has weight $2 \cdot 2^{k-1} = 2^k$. If $c_{k+2} = 1$, then $\mathbf{v}$ has the form

$$\mathbf{v} = \begin{bmatrix}\mathbf{u}\\ \mathbf{u}\end{bmatrix} + \begin{bmatrix}\mathbf{0}\\ \mathbf{1}\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \mathbf{u} + \mathbf{1}\end{bmatrix} = \begin{bmatrix}\mathbf{u}\\ \overline{\mathbf{u}}\end{bmatrix}$$

where $\mathbf{u}$ is in $R_k$. Since

$$w(\overline{\mathbf{u}}) = 2^k - w(\mathbf{u})$$

(why?), we have

$$w(\mathbf{v}) = w(\mathbf{u}) + w(\overline{\mathbf{u}}) = 2^k$$

as required. This completes the induction, and we conclude that the theorem is true for all $n \geq 1$.

As noted, *Mariner 9* required a code with $64 = 2^6$ vectors. By Theorem 8.5, the Reed-Muller code $R_5$ has dimension 6 over $\mathbb{Z}_2$. As you will see in the next section, it is also capable of detecting and correcting multiple errors. That is why the Reed-Muller code was the one that NASA used for the transmission of the *Mariner* photographs. Exercises 13–16 explore further aspects of this important class of codes.

## Exercises 8.4

### Linear Codes

*Which of the codes in Exercises 1–8 are linear codes?*

**1.** $C = \left\{\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$

**2.** $C = \left\{\begin{bmatrix}0\\0\end{bmatrix}, \begin{bmatrix}1\\1\end{bmatrix}\right\}$

**3.** $C = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}\right\}$

**4.** $C = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}\right\}$

**5.** $C = \left\{\begin{bmatrix}0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\1\end{bmatrix}, \begin{bmatrix}0\\0\\1\end{bmatrix}\right\}$

**6.** $C = \left\{\begin{bmatrix}0\\0\\0\\0\end{bmatrix}, \begin{bmatrix}1\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\1\end{bmatrix}, \begin{bmatrix}1\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\1\end{bmatrix}, \begin{bmatrix}1\\1\\1\\1\end{bmatrix}\right\}$

**7.** The even parity code $E_n$ (See Exercise 18 in Section 8.3.)

**8.** The odd parity code $O_n$ consisting of all vectors in $\mathbb{Z}_2^n$ with odd weight

**9.** Find the other two bases for the code $C_1$ in Example 8.13.

**10.** **(a)** If a (9, 4) linear code has generator matrix $G$ and parity check matrix $P$, what are the dimensions of $G$ and $P$?
**(b)** Repeat part (a) for an $(n, k)$ linear code.

**11.** For a linear code $C$, show that $(C^{\perp})^{\perp} = C$ without using matrices.

**12.** If $C$ is an $(n, k)$ linear code that is self dual, prove that $n$ must be even. [*Hint:* Use the analogue in $\mathbb{Z}_2^n$ of Theorem 5.13.]

**13.** Write out the vectors in the Reed-Muller code $R_3$.

**14.** Define a family of matrices inductively as follows: $G_0 = [1]$ and, for $n \geq 1$,

$$G_n = \left[\begin{array}{c|c} G_{n-1} & \mathbf{0} \\ \hline G_{n-1} & \mathbf{1} \end{array}\right]$$

where $\mathbf{0}$ is a zero vector and $\mathbf{1}$ is a vector consisting entirely of ones.
**(a)** Write out $G_1$, $G_2$, and $G_3$.
**(b)** Using induction, prove that for all $n \geq 0$, $G_n$ is a generator matrix for the Reed-Muller code $R_n$.

**15.** Find a parity check matrix for $R_2$.

**16.** Find a parity check matrix for $R_3$.

**17.** Prove that, for a linear code $C$, either all the code vectors have even weight or exactly half of them do. [*Hint:* Let $E$ be the set of vectors in $C$ with even weight and $O$ the set of vectors in $C$ with odd weight. If $O$ is not empty, let $\mathbf{c}_o$ be in $O$ and consider $O' = \{\mathbf{c}_o + \mathbf{e} : \mathbf{e} \text{ in } E\}$. Show that $O' = O$.]

## 8.5 The Minimum Distance of a Code

Consider the triple repetition code

$$C = \{\mathbf{c}_0, \mathbf{c}_1\} = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \right\}$$

If one or two errors occur in the transmission of either of these code vectors, the resulting vector cannot be another vector in $C$. So $C$ can detect up to two errors. For example, if errors occur in the first and second entries when $\mathbf{c}_0$ is transmitted, then the vector

$$\mathbf{c}' = \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}$$

is received. However, the receiver has no way of correcting the error, since $\mathbf{c}'$ would also result if a single error occurred during the transmission of $\mathbf{c}_1$. But any *single* error can be corrected, since the resulting vector can have arisen in only one way. For example, if

$$\mathbf{c}'' = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

is received and we know that at most one error has occurred, then the original vector must have been $\mathbf{c}_0$, since $\mathbf{c}''$ cannot arise from $\mathbf{c}_1$ via a single error.

We will now generalize these ideas. As you will see, the notion of Hamming distance plays a crucial role in the definition.

**Definition** Let $C$ be a (binary) code. The ***minimum distance*** of $C$ is the smallest Hamming distance between any two distinct vectors in $C$. That is,

$$d(C) = \min\{d_H(\mathbf{x}, \mathbf{y}) : \mathbf{x} \neq \mathbf{y} \text{ in } C\}$$

Clearly, the minimum distance of the triple repetition code $C$ above is 3.

**Example 8.14** Find the minimum distance of the code

$$C = \{\mathbf{c}_0, \mathbf{c}_1, \mathbf{c}_2, \mathbf{c}_3\}$$

where
$$\mathbf{c}_0 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \quad \mathbf{c}_1 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \quad \mathbf{c}_2 = \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \quad \mathbf{c}_3 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

**Solution** We need to compute the Hamming distance between each pair of distinct vectors. [There are four vectors, so there are $\binom{4}{2} = 6$ pairs.] We find that:

$$\begin{array}{lll} d_H(\mathbf{c}_0, \mathbf{c}_1) = 2 & d_H(\mathbf{c}_0, \mathbf{c}_2) = 2 & d_H(\mathbf{c}_0, \mathbf{c}_3) = 4 \\ d_H(\mathbf{c}_1, \mathbf{c}_2) = 4 & d_H(\mathbf{c}_1, \mathbf{c}_3) = 2 & d_H(\mathbf{c}_2, \mathbf{c}_3) = 2 \end{array}$$

Therefore, $d(C) = 2$.

It is possible to picture the notions of minimum distance and error correction geometrically. In the case of the triple repetition code $C$, we have a subset (actually, a subspace) of $\mathbb{Z}_2^3$. We can represent the vectors in $\mathbb{Z}_2^3$ as the vertices of a unit cube, as shown in Figure 8.4(a). The Hamming distance between any two vectors $\mathbf{x}$ and $\mathbf{y}$ is just the number of edges in a shortest path from $\mathbf{x}$ to $\mathbf{y}$. The code $C$ corresponds to two of these vertices, $\mathbf{c}_0$ and $\mathbf{c}_1$. The fact that $d(C) = 3$ corresponds to the fact that $\mathbf{c}_0$ and $\mathbf{c}_1$ are three units apart, as shown in Figure 8.4(b). If a received vector $\mathbf{x}$ is within one unit of either of these code vectors and we know that at most one error has occurred, we can correctly decode $\mathbf{x}$ as the *nearest* code vector. In Figure 8.4(b), $\mathbf{x}$ would be decoded as $\mathbf{c}_0$, and $\mathbf{y}$ would be decoded as $\mathbf{c}_1$. This agrees with the fact that $C$ can correct single but not double errors.

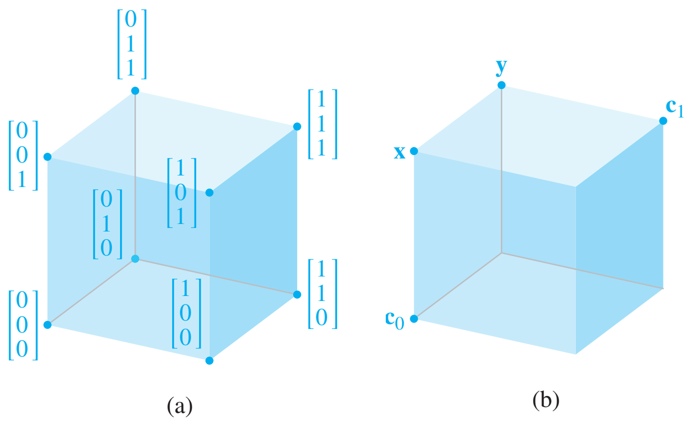


**Figure 8.4**

In Exercise 13, you are asked to draw a picture that illustrates the situation in Example 8.14. In general, we cannot draw pictures of $\mathbb{Z}_2^n$, but a Euclidean analogy is helpful. If a code can correct up to $k$ errors, think of the code vectors as the centers

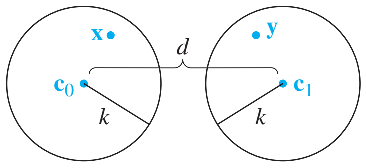


**Figure 8.5**

of spheres of radius $k$. The code vectors themselves are separated by at least $d$ units. Then, if a received vector $\mathbf{x}$ is inside one of these spheres, it will be decoded as the vector corresponding to the center of that sphere. In Figure 8.5, $\mathbf{x}$ will be decoded as $\mathbf{c}_0$. This process is known as ***nearest neighbor decoding.***

Figure 8.5 suggests that if a code is able to correct $k$ errors, then the "spheres" centered at the code vectors cannot touch or overlap; that is, we must have $d > 2k$. This turns out to be correct, as we now make precise. A code is said to ***detect k errors*** if, for each code vector $\mathbf{c}$ and each vector $\mathbf{c}'$ obtained by changing up to $k$ of the entries of $\mathbf{c}$, $\mathbf{c}'$ is not a code vector. A code is said to ***correct k errors*** if, for each code vector $\mathbf{c}$ and each vector $\mathbf{c}'$ obtained by changing up to $k$ of the entries of $\mathbf{c}$, nearest neighbor decoding of $\mathbf{c}'$ produces $\mathbf{c}$.

**Theorem 8.6** Let $C$ be a (binary) code with minimum distance $d$.

a. $C$ detects $k$ errors if and only if $d \geq k + 1$.
b. $C$ corrects $k$ errors if and only if $d \geq 2k + 1$.

**Proof** (a) Assume that $d \geq k + 1$ and let $\mathbf{c}$ be a vector in $C$. If up to $k$ errors are introduced into $\mathbf{c}$, then the resulting vector $\mathbf{c}'$ has the property that $d_H(\mathbf{c}, \mathbf{c}') \leq k$. But then $\mathbf{c}'$ cannot be a code vector, since if it were, we would have

$$k + 1 \leq d \leq d_H(\mathbf{c}, \mathbf{c}') \leq k$$

which is impossible.

Conversely, if $C$ can detect up to $k$ errors, then the minimum distance between any two code vectors must be greater than $k$. (Why?) It follows that $d \geq k + 1$.

(b) Assume that $d \geq 2k + 1$ and let $\mathbf{c}$ be a vector in $C$. As in the proof of property (a), let $\mathbf{c}'$ be a vector such that $d_H(\mathbf{c}, \mathbf{c}') \leq k$. Let $\mathbf{b}$ be another vector in $C$. Then $d_H(\mathbf{c}, \mathbf{b}) \geq d \geq 2k + 1$, so, by the Triangle Inequality,

$$d_H(\mathbf{c}, \mathbf{c}') + d_H(\mathbf{c}', \mathbf{b}) \geq d_H(\mathbf{c}, \mathbf{b}) \geq 2k + 1$$

Therefore,

$$d_H(\mathbf{c}', \mathbf{b}) \geq 2k + 1 - d_H(\mathbf{c}, \mathbf{c}') \geq 2k + 1 - k = k + 1 > d_H(\mathbf{c}', \mathbf{c})$$

So $\mathbf{c}'$ is closer to $\mathbf{c}$ than to $\mathbf{b}$, and nearest neighbor decoding correctly decodes $\mathbf{c}'$ as $\mathbf{c}$.

Conversely, assume that $C$ can correct up to $k$ errors. We will show that if $d < 2k + 1$ (i.e., $d \leq 2k$), then we obtain a contradiction. To do this, we will find a code vector $\mathbf{c}$ and a vector $\mathbf{c}'$ such that $d_H(\mathbf{c}, \mathbf{c}') \leq k$ yet nearest neighbor decoding decodes $\mathbf{c}'$ as the *wrong* code vector $\mathbf{b} \neq \mathbf{c}$.

Let $\mathbf{b}$ and $\mathbf{c}$ be any code vectors in $C$ such that

$$d_H(\mathbf{b}, \mathbf{c}) = d \leq 2k$$

There is no harm in assuming that these $d$ errors occur in the first $d$ entries of $\mathbf{b}$. (Otherwise, we can just permute the entries of all the vectors until this is true.) Assuming that the code vectors in $C$ have length $n$, we construct a vector $\mathbf{c}'$ in $\mathbb{Z}_2^n$ as follows. Make $\mathbf{c}'$ agree with $\mathbf{b}$ in the first $k$ entries, agree with $\mathbf{c}$ in the next $d - k$ entries (why is $d \geq k$?), and agree with both $\mathbf{b}$ and $\mathbf{c}$ in the last $n - d$ entries. In other words, the entries of $\mathbf{c}'$ satisfy

$$c_i' = \begin{cases} b_i \neq c_i & \text{if } i = 1, \dots, k \\ c_i \neq b_i & \text{if } i = k + 1, \dots, d \\ b_i = c_i & \text{if } i = d + 1, \dots, n \end{cases}$$

Now $d_H(\mathbf{c}, \mathbf{c}') = k$ and $d_H(\mathbf{c}', \mathbf{b}) = d - k \le k$. (Why?) Therefore, $d_H(\mathbf{c}', \mathbf{b}) \le d_H(\mathbf{c}', \mathbf{c})$, so either we have equality and it is impossible to decide whether $\mathbf{c}'$ should be decoded as $\mathbf{b}$ or $\mathbf{c}$ or the inequality is strict and $\mathbf{c}'$ will be incorrectly decoded as $\mathbf{b}$. In either case, we have shown that $C$ cannot correct $k$ errors, which contradicts our hypothesis. We conclude that $d \ge 2k + 1$.

Some books call such a code an $(n, 2^k, d)$ code or, more generally, an $(n, M, d)$ code, where $n$ is the length of the vectors, $M$ is the number of code vectors, and $d$ is the minimum distance.

In the case of a linear code, we have the following notation: If an $(n, k)$ linear code has minimum distance $d$, we refer to it as an ***(n, k, d)* code.** For example, the code in Example 8.14 is a (4, 2, 2) code. Linear codes have the advantage that their minimum distance can be easily determined. In Exercise 14, you are asked to show that the minimum distance of a linear code is the same as the minimum weight of a nonzero code vector. It is also possible to determine $d(C)$ by examining a parity check matrix for $C$.

**Theorem 8.7** Let $C$ be an $(n, k)$ linear code with parity check matrix $P$. Then the minimum distance of $C$ is the smallest integer $d$ for which $P$ has $d$ linearly dependent columns.

**Proof** Assume that $d(C) = d$. The parity check matrix $P$ is an $(n - k) \times n$ matrix with the property that, for any vector $\mathbf{x}$ in $\mathbb{Z}_2^n$, $P\mathbf{x} = \mathbf{0}$ if and only if $\mathbf{x}$ is in $C$. As you will be asked to show in Exercise 14, $C$ contains a vector $\mathbf{c}$ of weight $d$. Then $P\mathbf{c}$ is a linear combination of exactly $d$ columns of $P$. But, since $P\mathbf{c} = \mathbf{0}$, this implies that some set of $d$ columns of $P$ is linearly dependent. On the other hand, suppose some set of $d - 1$ columns of $P$ is linearly dependent—say,

$$\mathbf{p}_{i_1} + \mathbf{p}_{i_2} + \cdots + \mathbf{p}_{i_{d-1}} = \mathbf{0}$$

Let $\mathbf{x}$ be a vector in $\mathbb{Z}_2^n$ with 1s in positions $i_1, \ldots, i_{d-1}$ and zeros elsewhere. Then $\mathbf{x}$ is a vector of weight $d - 1$ such that $P\mathbf{x} = \mathbf{0}$. Hence, $\mathbf{x}$ is a code vector of weight $d - 1 < d = d(C)$. This is impossible, by Exercise 14, so we deduce that rank $(P) = d - 1$.

Conversely, assume that any $d - 1$ columns of $P$ are linearly independent but some set of $d$ columns of $P$ is linearly dependent. Since $P\mathbf{x}$ is a linear combination of those columns of $P$ corresponding to the positions of the 1s in $\mathbf{x}$, $P\mathbf{x} \ne \mathbf{0}$ for any vector $\mathbf{x}$ of weight $d - 1$ or less. Therefore, there are no nonzero code vectors of weight less than $d$. But some set of $d$ columns of $P$ is linearly dependent, so there exists a vector $\mathbf{x}$ of weight $d$ such that $P\mathbf{x} = \mathbf{0}$. Hence, this $\mathbf{x}$ is a code vector of weight $d$. By Exercise 14 again, we deduce that $d(C) = d$.

**Example 8.15** Show that the Hamming codes all have minimum distance 3.

**Solution** Recall that the $(n, k)$ Hamming code has an $(n - k) \times n$ parity check matrix $P$ whose columns are all of the nonzero vectors of $\mathbb{Z}_2^{n-k}$, arranged so that the identity matrix occupies the last $n - k$ columns. For example, the (7, 4) Hamming code has parity check matrix

$$P = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}$$

We can always find three linearly dependent columns: Just take the columns corresponding to $\mathbf{e}_1$, $\mathbf{e}_2$, and $\mathbf{e}_1 + \mathbf{e}_2$. (In the matrix on the previous page, these would be columns 5, 6, and 1, respectively.) But any two columns are linearly independent. By Theorem 8.7, this means the Hamming codes have minimum distance 3.

Example 8.15, combined with Theorem 8.6, tells us that the Hamming codes are all single error–correcting. The other major type of linear code that we have considered is the family of Reed-Muller codes. These are capable of correcting many errors, which is one of the reasons they were chosen to transmit photographs from space.

**Example 8.16** Show that the Reed-Muller code $R_n$ has minimum distance $2^{n-1}$ for $n \ge 1$.

**Solution** By Theorem 8.5, every vector in $R_n$ except $\mathbf{0}$ and $\mathbf{1}$ has weight $2^{n-1}$. Since $\mathbf{1}$ has weight $2^n$, this means that the minimum weight of a nonzero code vector in $R_n$ is $2^{n-1}$. Hence, $d(R_n) = 2^{n-1}$, by Exercise 14.

## Exercises 8.5

### The Minimum Distance of a Code

*Find the minimum distance of the codes in Exercises 1–6.*

**1.** $C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix} \right\}$

**2.** $C = \left\{ \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} \right\}$

**3.** The even parity code $E_n$

**4.** The $n$-times repetition code $\text{Rep}_n$

**5.** The code with parity check matrix $P = [I \mid A]$, where

$$A = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

**6.** The code with parity check matrix

$$P = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 \end{bmatrix}$$

*In Exercises 7 and 8, compute the minimum distance of the code C and decode the vectors* **u**, **v**, *and* **w** *using nearest neighbor decoding.*

**7.** $C = \left\{ \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} \right\}, \mathbf{u} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix},$

$\mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}$

**8.** $C$ has generator matrix

$$G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \mathbf{u} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix}, \mathbf{v} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \mathbf{w} = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix}$$

*In Exercises 9–12, construct a linear $(n, k, d)$ code or prove that no such code exists.*

**9.** $n = 8, k = 1, d = 8$ **10.** $n = 8, k = 2, d = 8$

**11.** $n = 8, k = 5, d = 5$ **12.** $n = 8, k = 4, d = 4$

**13.** Draw a picture (similar to Figure 8.4) to illustrate Example 8.14.

**14.** Let $C$ be a linear code. Show that the minimum distance of $C$ is equal to the minimum weight of a nonzero code vector.

**15.** Show that $d - 1 \leq n - k$ for any linear $(n, k, d)$ code.

**16.** Let $C$ be a linear $(n, k, d)$ code with parity check matrix $P$. Prove that $d = n - k + 1$ if and only if every $n - k$ columns of $P$ are linearly independent.