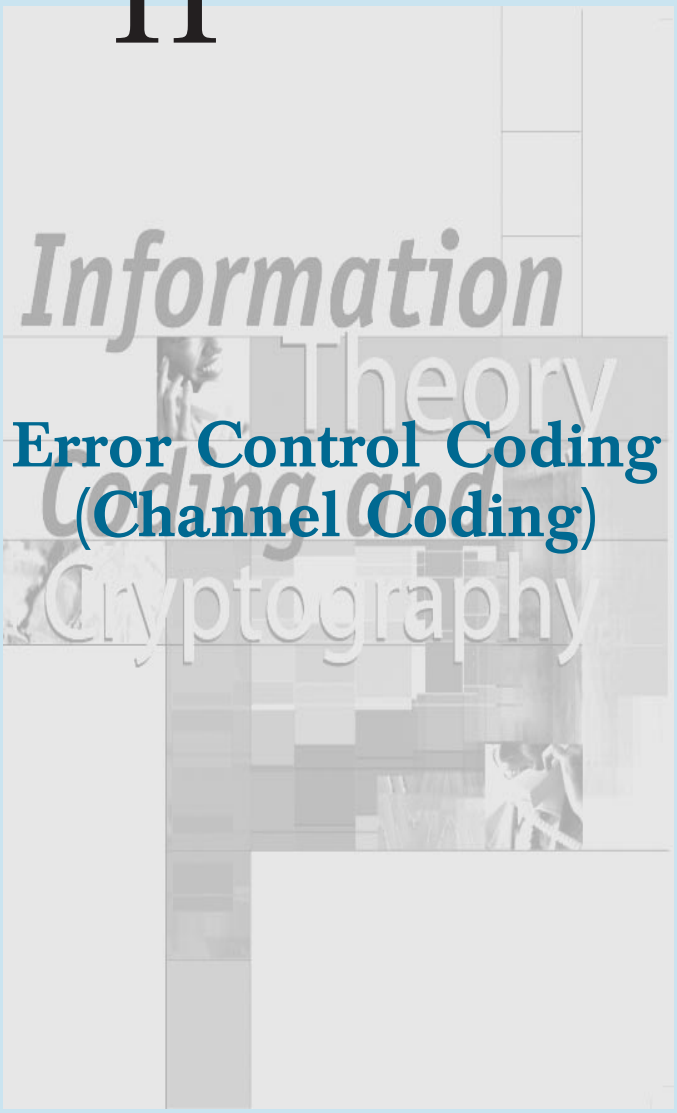
PART II



Error Control Coding (Channel Coding)

3

Linear Block Codes for Error Correction

Mathematics is an interesting intellectual sport but it should not be allowed to stand in the way of obtaining sensible information about physical processes.
Richard W. Hamming

3.1 INTRODUCTION TO ERROR CORRECTING CODES

In this *age of information*, there is increasing need not only for speed, but also for accuracy in the storage, retrieval, and transmission of data. The channels over which messages are transmitted are often imperfect. Machines do make errors, and their non-man-made mistakes can turn otherwise flawless programming into worthless, even dangerous, trash. Just as architects design buildings that will stand even through an earthquake, their computer counterparts have come up with sophisticated techniques capable of counteracting digital manifestations of Murphy's Law ("If anything can go wrong, it will go"). **Error Correcting Codes** are a kind of safety net– the mathematical insurance against the vagaries of an imperfect digital world.

Error Correcting Codes, as the name suggests, are used for correcting errors when messages are transmitted over a noisy channel or stored data is retrieved. The physical medium through which the messages are transmitted is called a channel (e.g. a telephone line, a satellite link, a wireless channel used for mobile communications etc.). Different kinds of channels are

prone to different kinds of noise, which corrupt the data being transmitted. The noise could be caused by lightning, human errors, equipment malfunction, voltage surge etc. Because these error correcting codes try to overcome the detrimental effects of noise in the channel, the encoding procedure is also called *Channel Coding*. Error control codes are also used for accurate transfer of information from one place to another, for example storing data and reading it from a compact disc (CD). In this case, the error could be due to a scratch on the surface of the CD. The error correcting coding scheme will try to recover the original data from the corrupted one.

The basic idea behind error correcting codes is to add some redundancy in the form of extra symbol to a message prior to its transmission through a noisy channel. This redundancy is added in a controlled manner. The encoded message when transmitted might be corrupted by noise in the channel. At the receiver, the original message can be recovered from the corrupted one if the number of errors are within the limit for which the code has been designed. The block diagram of a digital communication system is illustrated in Fig. 3.1. Note that the most important block in the figure is that of noise, without which there will be no need for the channel encoder.

Example 3.1 Let us see how redundancy combats the effects of noise. The normal language that we use to communicate (say, English) has a lot of redundancy built into it. Consider the following sentence.

CODNG THEORY IS AN INTRSTNG SUBJCT.

As we can see, there are a number of errors in this sentence. However, due to familiarity with the language we may guess the original text to have read:

CODING THEORY IS AN INTERESTING SUBJECT.

What we have just used is an error correcting strategy that makes use of the in-built redundancy in English language to reconstruct the original message from the corrupted one.

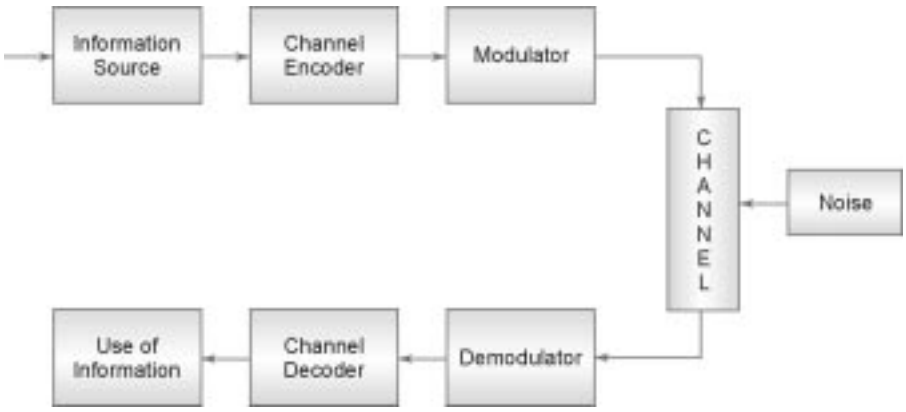


Fig. 3.1 *Block Diagram (and the principle) of a Digital Communication System. Here the Source Coder/Decoder Block has not been shown.*

The objectives of a good error control coding scheme are

(i) error correcting capability in terms of the number of errors that it can rectify
(ii) fast and efficient encoding of the message,
(iii) fast and efficient decoding of the received message
(iv) maximum transfer of information bits per unit time (i.e., fewer overheads in terms of redundancy).

The first objective is the primary one. In order to increase the error correcting capability of a coding scheme one must introduce more redundancies. However, increased redundancy leads to a slower rate of transfer of the actual information. Thus the objectives (i) and (iv) are not totally compatible. Also, as the coding strategies become more complicated for correcting larger number of errors, the objectives (ii) and (iii) also become difficult to achieve.

In this chapter, we shall first learn about the basic definitions of error control coding. These definitions, as we shall see, would be used throughout this book. The concept of **Linear Block Codes** will then be introduced. **Linear Block Codes** form a very large class of useful codes. We will see that it is very easy to work with the matrix description of these codes. In the later part of this chapter, we will learn how to efficiently decode these **Linear Block Codes**. Finally, the notion of perfect codes and optimal linear codes will be introduced.

3.2 BASIC DEFINITIONS

Given here are some basic definitions, which will be frequently used here as well as in the later chapters.

Definition 3.1 A **Word** is a sequence of symbols.

Definition 3.2 A **Code** is a set of vectors called **Codewords**.

Definition 3.3 The **Hamming Weight** of a **Codeword** (or any vector) is equal to the number of nonzero elements in the codeword. The Hamming Weight of a codeword $\boldsymbol{c}$ is denoted by $w(\boldsymbol{c})$. The **Hamming Distance** between two codewords is the number of places the codewords differ. The Hamming Distance between two codewords $\boldsymbol{c}_1$ and $\boldsymbol{c}_2$ is denoted by $d(\boldsymbol{c}_1, \boldsymbol{c}_2)$. It is easy to see that $d(\boldsymbol{c}_1, \boldsymbol{c}_2) = w(\boldsymbol{c}_1 - \boldsymbol{c}_2)$.

Example 3.2 Consider a code $\boldsymbol{C}$ with two code words = {0100, 1111} with Hamming Weight w (0100) = 1 and w (1111) = 4. The Hamming Distance between the two codewords is 3 because they differ at the 1st, 3rd and 4th places. Observe that w (0100 – 1111) = w (1011) = 3 = d (0100, 1111).

Example 3.3 For the code $\boldsymbol{C}$ = {01234, 43210}, the Hamming Weight of each codeword is 4 and the Hamming Distance between the codewords is 4 (because only the 3rd component of the two codewords are identical while they differ at 4 places).

Definition 3.4 A **Block Code** consists of a set of fixed length codewords. The fixed length of these codewords is called the **Block Length** and is typically denoted by n. Thus, a code of blocklength n consists of a set of codewords having n **components**.

A block code of size M defined over an alphabet with q symbols is a set of M q-ary sequences, each of length n. In the special case that $q = 2$, the symbols are called bits and the code is said to be a binary code. Usually, $M = q^k$ for some integer k, and we call such a code an (n, k) code.

Example 3.4 The code $\boldsymbol{C}$ = {00000, 10100, 11110, 11001} is a block code of block length equal to 5. This code can be used to represent two bit binary numbers as follows

Uncoded bits	Codewords
00	00000
01	10100
10	11110
11	11001

Here $M = 4$, $k = 2$ and $n = 5$. Suppose we have to transmit a sequence of 1's and 0's using the above coding scheme. Let's say that the sequence to be encoded is 1 0 0 1 0 1 0 0 1 1 ... The first step is to break the sequence in groups of two bits (because we want to encode two bits at a time). So we partition as follows

10 01 01 00 11 ...

Next, replace each block by its corresponding codeword.

11110 10100 10100 00000 11001 ...

Thus 5 bits (coded) are sent for every 2 bits of uncoded message. It should be observed that for every 2 bits of information we are sending 3 extra bits (redundancy).

Definition 3.5 The **Code Rate** of an (n, k) code is defined as the ratio (k/n), and denotes the fraction of the codeword that consists of the information symbols.

Code rate is always less than unity. The smaller the code rate, the greater the redundancy, i.e., more of redundant symbols are present *per information symbol* in a codeword. A code with greater redundancy has the potential to detect and correct more of symbols in error, but reduces the actual rate of transmission of information.

Definition 3.6 The **minimum distance** of a code is the minimum Hamming distance between any two codewords. If the code $\boldsymbol{C}$ consists of the set of codewords $\{\boldsymbol{c}_i, i = 0, 1, ..., M-1\}$ then the minimum distance of the code is given by $d^* = \min d(\boldsymbol{c}_i, \boldsymbol{c}_j)$, $i \neq j$. An (n, k) code with minimum distance d^* is sometimes denoted by (n, k, d^*).

Definition 3.7 The **minimum weight** of a code is the smallest weight of any non-zero codeword, and is denoted by w^*.

Theorem 3.1 For a linear code the minimum distance is equal to the minimum weight of the code, i.e., $d^* = w^*$.

Intuitive proof: The distance d_{ij} between any two codewords c_i and c_j is simply the weight of the codeword formed by $c_i - c_j$. Since the code is linear, the difference between two codewords results in another valid codeword. Thus, the minimum weight of a non-zero codeword will reflect the minimum distance of the code.

Definition 3.8 A **linear code** has the following properties:

(i) The sum of two codewords belonging to the code is also a codeword belonging to the code.

(ii) The all-zero word is always a codeword.

(iii) The minimum Hamming distance between two codewords of a linear code is equal to the minimum weight of any non-zero codeword, i.e., $d^* = w^*$.

Note that if the sum of two codewords is another codeword, the difference of two codewords will also yield a valid codeword. For example, if $\boldsymbol{c}_1$, $\boldsymbol{c}_2$ and $\boldsymbol{c}_3$ are valid codewords such that $\boldsymbol{c}_1 + \boldsymbol{c}_2 = \boldsymbol{c}_3$ then $\boldsymbol{c}_3 - \boldsymbol{c}_1 = \boldsymbol{c}_2$. Hence it is obvious that the all-zero codeword must always be a valid codeword for a linear block code (self-subtraction of a codeword).

Example 3.5 The code $\boldsymbol{C}$ = {0000, 1010, 0101, 1111} is a linear block code of block length n = 4. Observe that all the ten possible sums of the codewords

0000 + 0000 = 0000, 0000 + 1010 = 1010, 0000 + 0101 = 0101,

0000 + 1111 = 1111, 1010 + 1010 = 0000, 1010 + 0101 = 1111,

1010 + 1111 = 0101, 0101 + 0101 = 0000, 0101 + 1111 = 1010 and

1111 + 1111 = 0000

are in $\boldsymbol{C}$ and the all-zero codeword is in $\boldsymbol{C}$. The minimum distance of this code is $d^* = 2$. In order to verify the minimum distance of this linear w code we can determine the distance between all pairs of codewords (which is $\binom{4}{2} = 6$ in number):

d (0000, 1010) = 2, d (0000, 0101) = 2, d (0000, 1111) = 4

d (1010, 0101) = 4, d (1010, 1111) = 2, d (0101, 1111) = 2

We observe that the minimum distance of this code is 2.

Note that the code given in Example 3.4 is not linear because 1010 + 1111 = 0101, which is not a valid codeword. Even though the all-zero word is a valid codeword, it does not guarantee linearity. **The presence of an all-zero codeword is thus a necessary but not a sufficient condition for linearity**.

In order to make the error correcting codes easier to use, understand and analyze, it is helpful to impose some basic algebraic structure on them. As we shall soon see, it is useful to have an alphabet werein it is easy to carry out basic mathematical operations such as add, subtract, multiply and divide.

Definition 3.9 A field $\boldsymbol{F}$ is a set of elements with two operations + (addition) and . (multiplication) satisfying the following properties

(i) $\boldsymbol{F}$ is closed under + and ., i.e., $a + b$ and $a \cdot b$ are in $\boldsymbol{F}$ if a and b are in $\boldsymbol{F}$.

For all a, b and c in $\boldsymbol{F}$, the following hold:

(ii) Commutative laws: $a + b = b + a$, $a \cdot b = b.a$

(iii) Associative laws: $(a + b) + c = a + (b + c)$, $a \cdot (b \cdot c) = (a \cdot b) \cdot c$

(iv) Distributive law: $a \cdot (b + c) = a \cdot b + a \cdot c$

Further, identity elements 0 and 1 must exist in $\boldsymbol{F}$ satisfying:

(v) $a + 0 = a$

(vi) $a \cdot 1 = a$

(vii) For any a in $\boldsymbol{F}$, there exists an additive inverse $(-a)$ such that $a + (-a) = 0$.

(viii) For any a in $\boldsymbol{F}$, there exists an multiplicative inverse (a^{-1}) such that $a \cdot a^{-1} = 1$.

The above properties are true for fields with both finite as well as infinite elements. A field with a finite number of elements (say, q) is called a Galois Field (pronounced Galva Field) and is denoted by $GF(q)$. If only the first seven properties are satisfied, then it is called a **ring**.

Example 3.6 Consider GF (4) with 4 elements {0, 1, 2, 3}. The addition and multiplication tables for $GF(4)$ are

+	**0**	**1**	**2**	**3**
0	0	1	2	3
1	1	0	3	2
2	2	3	0	1
3	3	2	1	0

.	**0**	**1**	**2**	**3**
0	0	0	0	0
1	0	1	2	3
2	0	2	3	1
3	0	3	1	2

It should be noted here that the addition in $GF(4)$ is not modulo 4 addition.

Let us define a vector space, $GF(q)^n$, which is a set of n-tuples of elements from $GF(q)$. Linear block codes can be looked upon as a set of n-tuples (vectors of length n) over $GF(q)$ such that the sum of two codewords is also a codeword, and the product of any codeword by a field element is a codeword. Thus, a linear block code is a subspace of $GF(q)^n$.

Let $\boldsymbol{S}$ be a set of vectors of length n whose components are defined over $GF(q)$. The set of all linear combinations of the vectors of $\boldsymbol{S}$ is called the linear span of $\boldsymbol{S}$ and is denoted by $<\boldsymbol{S}>$. The linear span is thus a subspace of $GF(q)^n$, generated by $\boldsymbol{S}$. Given any subset $\boldsymbol{S}$ of $GF(q)^n$, it is possible to obtain a linear code $\boldsymbol{C} = <\boldsymbol{S}>$ generated by $\boldsymbol{S}$, consisting of precisely the following codewords:

(i) all-zero word,
(ii) all words in $\boldsymbol{S}$,
(iii) all linear combinations of two or more words in $\boldsymbol{S}$.

Example 3.7 Let $\boldsymbol{S}$ = {1100, 0100, 0011}. All possible linear combinations of $\boldsymbol{S}$ are 1100 + 0100 = 1000, 1100 + 0011 = 1111, 0100 + 0011 = 0111, 1100 + 0100 + 0011 = 1011.

Therefore, $\boldsymbol{C} = <\boldsymbol{S}> = \{0000, 1100, 0100, 0011, 1000, 1111, 0111, 1011\}$. The minimum distance of this code is $w(0100) = 1$.

Example 3.8 Let $\boldsymbol{S}$ = {12, 21} defined over $GF(3)$. The addition and multiplication tables of field $GF(3) = \{0, 1, 2\}$ are given by:

+	0	1	2
0	0	1	2
1	1	2	0
2	2	0	1

.	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

All possible linear combinations of 12 and 21 are:

12 + 21 = 00, 12 + 2(21) = 21, 2(12) + 21 = 12.
Therefore, $\boldsymbol{C} = <\boldsymbol{S}> = \{00, 12, 21, 00, 21, 12\} = \{00, 12, 21\}$.

3.3 MATRIX DESCRIPTION OF LINEAR BLOCK CODES

As we have observed earlier, any code $\boldsymbol{C}$ is a subspace of $GF(q)^n$. Any set of basis vectors can be used to generate the code space. We can, therefore, define a generator matrix, $\boldsymbol{G}$, the rows of which form the basis vectors of the subspace. The rows of $\boldsymbol{G}$ will be linearly independent. Thus, a linear combination of the rows can be used to generate the codewords of $\boldsymbol{C}$. The generator matrix will be a $k \times n$ matrix with rank k. Since the choice of the basis vectors is not unique, *the generator matrix is not unique for a given linear code.*

The generator matrix converts (encodes) a vector of length k to a vector of length n. Let the input vector (uncoded symbols) be represented by $\boldsymbol{i}$. The coded symbols will be given by

$$\boldsymbol{c} = \boldsymbol{iG} \tag{3.1}$$

where $\boldsymbol{c}$ is called the codeword and $\boldsymbol{i}$ is called the information word.

The generator matrix provides a concise and efficient way of representing a linear block code. The $n \times k$ matrix can generate q^k codewords. Thus, instead of having a large look-up table of q^k codewords, one can simply have a generator matrix. This provides an enormous saving in storage space for large codes. For example, for the binary (46, 24) code the total number of codewords are $2^{24} = 1{,}777{,}216$ and the size of the lookup table of codewords will be $n \times 2^k = 771{,}751{,}936$ bits. On the other hand if we use a generator matrix, the total storage requirement would be $n \times k = 46 \times 24 = 1104$ bits.

Example 3.9 Consider a generator matrix

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

$$\boldsymbol{c}_1 = [0\ 0]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [0\ 0\ 0], \qquad \boldsymbol{c}_2 = [0\ 1]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [0\ 1\ 0]$$

$$\boldsymbol{c}_3 = [1\ 0]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1\ 0\ 1], \qquad \boldsymbol{c}_4 = [1\ 1]\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = [1\ 1\ 1]$$

Therefore, this generator matrix generates the code $\boldsymbol{C} = \{000, 010, 101, 111\}$.

3.4 EQUIVALENT CODES

Definition 3.10 A **permutation** of a set $\boldsymbol{S} = \{x_1, x_2, \ldots, x_n\}$ is a one to one mapping from $\boldsymbol{S}$ to itself. A permutation can be denoted as follows

$$\begin{pmatrix} x_1 & x_2 & \cdots & x_n \\ \downarrow & \downarrow & \cdots & \downarrow \\ f(x_1) & f(x_2) & \cdots & f(x_n) \end{pmatrix} \tag{3.2}$$

Definition 3.11 Two q-ary codes are called **equivalent** if one can be obtained from the other by one or both operations listed below:

(i) permutation of the symbols appearing in a fixed position,

(ii) permutation of the positions of the code.

Suppose a code containing M codewords are displayed in the form of an $M \times n$ matrix, where the rows represent the codewords. The operation (i) corresponds to the re-labelling of the symbols appearing in a given column, and the operation (ii) represents the rearrangements of the colums of the matrix.

Example 3.10 Consider the ternary code (a code whose components $\in \{0, 1, 2\}$) of blocklength 3

$$\boldsymbol{C} = \begin{cases} 2 & 0 & 1 \\ 1 & 2 & 0 \\ 0 & 1 & 2 \end{cases}$$

If we apply the permutation $0 \to 2$, $2 \to 1$, $1 \to 0$ to column 2 and $1 \to 2$, $0 \to 1$, $2 \to 0$ to column 3 we obtain

$$\boldsymbol{C1} = \begin{cases} 2 & 2 & 2 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{cases}$$

The code $\boldsymbol{C1}$ is equivalent to a repetition code of length 3.

Note that the original code is not linear, but is equivalent to a linear code.

Definition 3.12 Two linear q-ary codes are called **equivalent** if one can be obtained from the other by one or both operations listed below:

(i) multiplication of the components by a non-zero scalar,

(ii) permutation of the positions of the code.

Note that in Definition 3.11 we have defined equivalent codes that are not necessarily linear.

Theorem 3.2 Two $k \times n$ matrices generate equivalent linear (n, k) codes over $GF(q)$ if one matrix can be obtained from the other by a sequence of the following operations:

(i) Permutation of rows

(ii) Multiplication of a row by a non scalar

(iii) Addition of a scalar multiple of one row to another

(iv) Permutation of columns

(v) Multiplication of any column by a non-zero scalar.

Proof The first three operations (which are just row operations) preserve the linear independence of the rows of the generator matrix. The operations merely modify the basis. The last two operations (which are column operations) convert the matrix to one which will produce an equivalent code.

Theorem 3.3 A generator matrix can be reduced to its **systematic form** (also called the standard form of the generator matrix) of the type $\boldsymbol{G} = [\boldsymbol{I} \mid \boldsymbol{P}]$ where $\boldsymbol{I}$ is a $k \times k$ identity matrix and $\boldsymbol{P}$ is a $k \times (n - k)$ matrix.

Proof The k rows of any generator matrix (of size $k \times n$) are linearly independent. Hence, by performing elementary row operations and column permutations it is possible to obtain an equivalent generator matrix in a row echelon form. This matrix will be of the form $[\boldsymbol{I} \mid \boldsymbol{P}]$.

Example 3.11 Consider the generator matrix of a (4, 3) code over $GF(3)$:

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

Let us represent the i^{th} row by r_i and the j^{th} column by r_j. Upon replacing r_3 by $r_3 - r_1 - r_2$ we get (note that in $GF(3)$, $-1 = 2$ and $-2 = 1$ because $1 + 2 = 0$, see table in Example 3.6)

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

Next we replace r_1 by $r_1 - r_3$ to obtain

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

Finally, shifting $c_4 \rightarrow c_1$, $c_1 \rightarrow c_2$, $c_2 \rightarrow c_3$ and $c_3 \rightarrow c_4$ we obtain the standard form of the generator matrix

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

3.5 PARITY CHECK MATRIX

One of the objectives of a good code design is to have fast and efficient encoding and decoding methodologies. So far we have dealt with the efficient generation of linear block codes using a generator matrix. Codewords are obtained simply by multiplying the input vector (uncoded word) by the generator matrix. Is it possible to detect a valid codeword using a similar concept? The answer is yes, and such a matrix is called the **Parity Check Matrix**, $\boldsymbol{H}$, for the given code. For a parity check matrix,

$$\boldsymbol{cH}^T = \mathbf{0} \tag{3.3}$$

where $\boldsymbol{c}$ is a valid codeword. Since $\boldsymbol{c} = \boldsymbol{iG}$, therefore, $\boldsymbol{iGH}^T = \mathbf{0}$. For this to hold true for all valid informat words we must have

$$\boldsymbol{GH}^T = \mathbf{0} \tag{3.4}$$

The size of the parity check matrix is $(n - k) \times n$. A parity check matrix provides a simple method of *detecting* whether an error has occurred or not. If the multiplication of the received word (at the receiver) with the transpose of $\boldsymbol{H}$ yields a non-zero vector, it implies that an error has occurred. This methodology, however, will fail if the errors in the transmitted codeword exceed the number of errors for which the coding scheme is designed. We shall soon find out that the non-zero product of $\boldsymbol{cH}^T$ might help us not only to detect but also to correct the errors under some conditions.

Suppose the generator matrix is represented in its systematic form $\boldsymbol{G} = [\boldsymbol{I} \mid \boldsymbol{P}]$. The matrix $\boldsymbol{P}$ is called the **Coefficient Matrix**. Then the parity check matrix will be defined as

$$\boldsymbol{H} = [-\boldsymbol{P}^T \mid \boldsymbol{I}], \tag{3.5}$$

where $\boldsymbol{P}^T$ represents the transpose of matrix $\boldsymbol{P}$. This is because

$$\boldsymbol{GH}^T = [\boldsymbol{I} \mid \boldsymbol{P}] \begin{bmatrix} -\boldsymbol{P} \\ \boldsymbol{I} \end{bmatrix} = \mathbf{0}. \tag{3.6}$$

Since the choice of a generator matrix is not unique for a code, *the parity check matrix will not be unique either*. Given a generator matrix $\boldsymbol{G}$, we can determine the corresponding parity check matrix and *vice versa*. Thus the parity check matrix $\boldsymbol{H}$ can be used to specify the code completely.

From Eq. (3.3) we observe that the vector $\boldsymbol{c}$ must have 1's in such positions that the corresponding rows of $\boldsymbol{H}^T$ add up to the zero vector $\mathbf{0}$. Now, we know that the number of 1's in a codeword pertains to its Hamming weight. *Hence, the minimum distance d^* of a linear block code is given by the minimum number of rows of $\boldsymbol{H}^T$ (or, the columns of $\boldsymbol{H}$) whose sum is equal to the zero vector.*

Example 3.12 For a (7, 4) linear block code the generator matrix is given by

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

the matrix $\boldsymbol{P}$ is given by $\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \end{bmatrix}$ and $\boldsymbol{P}^T$ is given by $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{bmatrix}$. Observing the fact that $-1 = 1$ for the case of binary, we can write the parity check matrix as

$$\boldsymbol{H} = [-\boldsymbol{P}^T \mid \boldsymbol{I}]$$

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

Note that the columns 1, 5 and 7 of the parity check matrix, $\boldsymbol{H}$, add up to the zero vector.

Hence, for this code, $d^* = 3$.

Theorem 3.4 The code $\boldsymbol{C}$ contains a nonzero codeword of Hamming weight w or less if and only if a linearly dependent set of w columns of $\boldsymbol{H}$ exist.

Proof Consider a codeword $\boldsymbol{c} \in \boldsymbol{C}$. Let the weight of $\boldsymbol{c}$ be w which implies that there are w non-zero components and $(n - w)$ zero components in $\boldsymbol{c}$. If we throw away the w zero components, then from the relation $\boldsymbol{C}\boldsymbol{H}^T = \boldsymbol{0}$ we can conclude that w columns of $\boldsymbol{H}$ are linearly dependent.

Conversely, if $\boldsymbol{H}$ has w linearly dependent columns, then a linear combination of at most w columns is zero. These w non-zero coefficients would define a codeword of weight w or less that satisfies $\boldsymbol{C}\boldsymbol{H}^T = \boldsymbol{0}$.

Definition 3.13 An (n, k) **systematic code** is one in which the first k symbols of the codeword of block length n are the information symbols themselves (i.e., the uncoded vector) and the remainder the $(n - k)$ symbols form the **parity symbols**.

Example 3.13 The following is a (5, 2) systematic code over $GF(3)$

S.No.	Information Symbols ($k = 2$)	Codewords ($n = 5$)
1.	00	00 000
2.	01	01 121
3.	02	02 220
4.	10	10 012
5.	11	11 221
6.	12	12 210
7.	20	20 020
8.	21	21 100
9.	22	22 212

Note that the total number of codewords is $3^k = 3^2 = 9$. Each codeword begins with the information symbols, and has three parity symbols at the end. The parity symbols for the information word 01 are 121 in the above table. A generator matrix in the systematic form (standard form) will generate a systematic code.

Theorem 3.5 The minimum distance (minimum weight) of an (n, k) linear code is bounded as follows

$$d^* \leq n - k + 1 \tag{3.7}$$

This is known as the **Singleton Bound**.

Proof We can reduce all linear block codes to their equivalent systematic forms. A systematic code can have one information symbol and $(n - k)$ parity symbols. At most all the parity symbols can be non-zero, resulting in the total weight of the codeword to be $(n - k) + 1$. Thus the weight of no codeword can exceed $n - k + 1$ giving the following definition of a maximum distance code.

Definition 3.14 A **Maximum Distance Code** satisfies $d^* = n - k + 1$.

Having familiarized ourselves with the concept of minimum distance of a linear code, we shall now explore how this minimum distance is related to the total number of errors the code can detect and possibly correct. So we move over to the receiver end and take a look at the methods of decoding a linear block code.

3.6 DECODING OF A LINEAR BLOCK CODE

The basic objective of channel coding is to detect and correct errors when messages are transmitted over a noisy channel. The noise in the channel randomly transforms some of the

symbols of the transmitted codeword into some other symbols. If the noise, for example, changes just one of the symbols in the transmitted codeword, the erroneous codeword will be at a Hamming distance of one from the original codeword. If the noise transforms t symbols (that is, t symbols in the codeword are in error), the Hamming distance of the received word will be at a Hamming distance of t from the originally transmitted codeword. Given a code, how many errors can it detect and how many can it correct? Let us first look at the detection problem.

An error will be detected as long as it does not transform one codeword into another valid codeword. If the minimum distance between the codewords is d^*, the weight of the error pattern must be d^* or more to cause a transformation of one codeword to another. Therefore, an (n, k, d^*) code will detect at least all nonzero error patterns of weight less than or equal to $(d^* - 1)$.

Moreover, there is at least one error pattern of weight d^* which will not be detected. This corresponds to the two codewords that are the closest. It may be possible that some error patterns of weight d^* or more are detected, but *all* error patterns of weight d^* will not be detected.

Example 3.14 For the code $\boldsymbol{C_1} = \{000, 111\}$ the minimum distance is 3. Therefore error patterns of weight 2 or 1 can be detected. This means that any error pattern belonging to the set {011, 101, 110, 001, 010, 100} will be detected by this code.

Next consider the code $\boldsymbol{C_2} = \{001, 110, 101\}$ with $d^* = 1$. Nothing can be said regarding how many errors this code can detect because $d^* - 1 = 0$. However, the error pattern 010 of weight 1 can be detected by this code. But it cannot detect all error patterns with weight one, e.g., the error vector 100 cannot be detected.

Next let us look at the problem of error correction. The objective is to make the best possible guess regarding the originally transmitted codeword on the basis of the received word. What would be a smart decoding strategy? Since only one of the valid codewords must have been transmitted, it is logical to conclude that a valid codeword nearest (in terms of Hamming distance) to the received word must have been actually transmitted. In other words, the codeword which resembles the received word most is assumed to be the one that was sent. This strategy is called the **Nearest Neighbour Decoding**, as we are picking the codeword nearest to the received word in terms of the Hamming distance.

It may be possible that more than one codeword is at the same Hamming distance from the received word. In that case the receiver can do one of the following:

(i) It can pick one of the equally distant neighbours randomly, or
(ii) request the transmitter to re-transmit.

To ensure that the received word (that has at most t errors) is closest to the original codeword, and farther from all other codewords, we must put the following condition on the minimum distance of the code

$$d^* \leq 2t + 1 \tag{3.8}$$

Graphically, the condition for correcting t errors or less can be visualized from Fig. 3.2. Consider the space of all q-ary n-tuples. Every q-ary vector of length n can be represented as a point in this space. Every codeword can thus be depicted as a point in this space, and all words at a Hamming distance of t or less would lie within the sphere centred at the codeword and with a radius of t. If the minimum distance of the code is d^*, and the condition $d^* \geq 2t + 1$ holds good, then none of these spheres would intersect. Any received vector (which is just a point) within a specific sphere will be closest to its centre (which represents a codeword) than any other codeword. We will call the spheres associated with each codeword its **Decoding Sphere**. Hence it is possible to decode the received vector using the 'nearest neighbour' method without ambiguity.

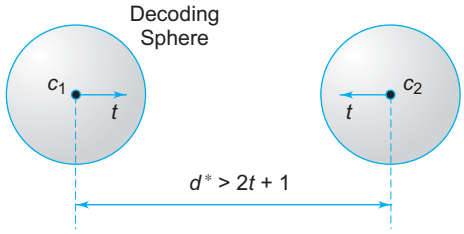


Fig. 3.2 *Decoding Spheres.*

Figure 3.2 shows words within the sphere of radius t and centred at $\boldsymbol{c}_1$ will be decoded as $\boldsymbol{c}_1$. For unambiguous decoding $d^* \geq 2t + 1$.

The condition $d^* \geq 2t + 1$ takes care of the worst case scenario. It may be possible, however, that the above condition is not met but it is still feasible to correct t errors as illustrated in the following example.

Example 3.15 Consider the code $\boldsymbol{C}$ = {00000, 01010, 10101, 11111}. The minimum distance $d^* = 2$. Suppose the codeword 11111 was transmitted and the received word is 11110, i.e., $t = 1$ (one error has occurred in the fifth component). Now,

$$d\,(11110, 00000) = 4,\ d\,(11110, 01010) = 2,$$
$$d\,(11110, 10101) = 3,\ d\,(11110, 11111) = 1.$$

Using the nearest neighbour decoding we can conclude that 11111 was transmitted. Even though a single error correction ($t = 1$) was done in this case, $d^* < 2t + 1 = 3$. So it is possible to correct

errors even when $d^* \geq 2t + 1$. However, in many cases a single error correction may not be possible with this code. For example, if 00000 was sent and 01000 was received,

$$d\,(01000, 00000) = 1,\ d\,(01000, 01010) = 1,$$
$$d\,(01000, 10101) = 4,\ d\,(01000, 11111) = 4.$$

In this case there cannot be a clear cut decision, and a coin will have to be flipped!

Definition 3.15 An **Incomplete Decoder** decodes only those received codewords that are clearly closest to one of the codewords. In the case of ambiguity, the decoder declares that the received word is unrecognizable. The receiver is then requested to re-transmit. A **Complete Decoder** decodes every received word, i.e., it tries to map every received word to some codeword, even if it has to make a guess. Example 3.16 was that of a Complete Decoder. Such decoders may be used when it is better to have a good guess rather than to have no guess at all. Most of the real life decoders are incomplete decoders. Usually they send a message back to the transmitter requesting them to re-transmit.

Definition 3.16 A receiver declares that an **erasure** has occurred (i.e., a received symbol has been erased) when the symbol is received ambiguously, or the presence of an interference is detected during reception.

Example 3.16 Consider a binary **Pulse Amplitude Modulation** (PAM) **Scheme** where 1 is represented by five volts and 0 is represented by zero volts. The noise margin is one volt, which implies that at the receiver:

if the received voltage is between 4 volts and 5 volts $\Rightarrow$ the bit sent is 1,

if the received voltage is between 0 volt and 1 volt $\Rightarrow$ the bit sent is 0,

if the received voltage is between 1 volt and 4 volts $\Rightarrow$ an erasure has occurred.

Thus if the receiver received 2.9 volts during a bit interval, it will declare that an erasure has occurred.

A channel can be prone both to errors and erasures. If in such a channel t errors and r erasures occur, the error correcting scheme should be able to compensate for the erasures as well as correct the errors. If r erasures occur, the minimum distance of the code will become $d^* - r$ in the worst case. This is because, the erased symbols have to be simply discarded, and if they

were contributing to the minimum distance, this distance will reduce. A simple example will illustrate the point. Consider the **repetition code** in which

$$0 \rightarrow 00000$$
$$1 \rightarrow 11111$$

Here $d^* = 5$. If $r = 2$, i.e., two bits get erased (let us say the first two), we will have

$$0 \rightarrow ??000$$
$$1 \rightarrow ??111$$

Now, the effective minimum distance $d_1^* = d^* - r = 3$.

Therefore, for a channel with t errors and r erasures, $d^* - r \geq 2t + 1$. Or,

$$d^* \geq 2t + r + 1 \tag{3.9}$$

For a channel which has no errors ($t = 0$), only r erasures,

$$d^* \geq r + 1 \tag{3.10}$$

Next let us give a little more formal treatment to the decoding procedure. Can we construct some mathematical tools to simplify the nearest neighbour decoding? Suppose the codeword $\boldsymbol{c} = c_1 c_2, \ldots, c_n$ is transmitted over a noisy channel. The noise in the channel changes some or all of the symbols of the codeword. Let the received vector be denoted by $\boldsymbol{v} = v_1 v_2, \ldots, v_n$. Define the **error vector** as

$$\boldsymbol{e} = \boldsymbol{v} - \boldsymbol{c} = v_1 v_2, \ldots, v_n - c_1 c_2, \ldots, c_n = e_1 e_2, \ldots, e_n \tag{3.11}$$

The decoder has to decide from the received vector, $\boldsymbol{v}$, which codeword was transmitted, or equivalently, it must determine the error vector, $\boldsymbol{e}$.

Definition 3.17 Let $\boldsymbol{C}$ be an (n, k) code over $GF(q)$ and $\boldsymbol{a}$ be any vector of length n. Then the set

$$\boldsymbol{a} + \boldsymbol{C} = \{\boldsymbol{a} + \boldsymbol{x} \mid \boldsymbol{x} \in \boldsymbol{C}\} \tag{3.12}$$

is called a **Coset** (or translate) of $\boldsymbol{C}$. $\boldsymbol{a}$ and $\boldsymbol{b}$ are said to be in the same coset if $(\boldsymbol{a} - \boldsymbol{b}) \in \boldsymbol{C}$.

Theorem 3.6 Suppose $\boldsymbol{C}$ is an (n, k) code over $GF(q)$. Then,

(i) every vector $\boldsymbol{b}$ of length n is in some coset of $\boldsymbol{C}$.
(ii) each coset contains exactly q^k vectors.
(iii) two cosets are either disjoint or coincide (partial overlap is not possible).
(iv) if $\boldsymbol{a} + \boldsymbol{C}$ is a coset of $\boldsymbol{C}$ and $\boldsymbol{b} \in \boldsymbol{a} + \boldsymbol{C}$, we have $\boldsymbol{b} + \boldsymbol{C} = \boldsymbol{a} + \boldsymbol{C}$.

Proof

(i) $\boldsymbol{b} = \boldsymbol{b} + 0 \in \boldsymbol{b} + \boldsymbol{C}$.
(ii) Observe that the mapping $\boldsymbol{C} \rightarrow \boldsymbol{a} + \boldsymbol{C}$ defined by $\boldsymbol{x} \rightarrow \boldsymbol{a} + \boldsymbol{x}$, for all $\boldsymbol{x} \in \boldsymbol{C}$ is a one-to-one mapping. Thus the cardinality of $\boldsymbol{a} + \boldsymbol{C}$ is the same as that of $\boldsymbol{C}$, which is equal to q^k.

(iii) Suppose the cosets $\boldsymbol{a} + \boldsymbol{C}$ and $\boldsymbol{a} + \boldsymbol{C}$ overlap, i.e., they have at least one vector in common.

Let $\boldsymbol{v} \in (\boldsymbol{a} + \boldsymbol{C}) \cap (\boldsymbol{b} + \boldsymbol{C})$. Thus, for some $\boldsymbol{x}, \boldsymbol{y} \in \boldsymbol{C}$,

$$\boldsymbol{v} = \boldsymbol{a} + \boldsymbol{x} = \boldsymbol{b} + \boldsymbol{y}.$$

Or, $$\boldsymbol{b} = \boldsymbol{a} + \boldsymbol{x} - \boldsymbol{y} = \boldsymbol{a} + \boldsymbol{z}, \text{ where } \boldsymbol{z} \in \boldsymbol{C}$$

(because, the difference of two codewords is also a codeword).

Thus, $$\boldsymbol{b} + \boldsymbol{C} = \boldsymbol{a} + \boldsymbol{C} + \boldsymbol{z} \text{ or } (\boldsymbol{b} + \boldsymbol{C}) \subset (\boldsymbol{a} + \boldsymbol{C}).$$

Similarly, it can be shown that $(\boldsymbol{a} + \boldsymbol{C}) \subset (\boldsymbol{b} + \boldsymbol{C})$. From these two we can conclude that $(\boldsymbol{b} + \boldsymbol{C}) = (\boldsymbol{a} + \boldsymbol{C})$.

(iv) Since $\boldsymbol{b} \in \boldsymbol{a} + \boldsymbol{C}$, it implies that $\boldsymbol{b} = \boldsymbol{a} + \boldsymbol{x}$, for some $\boldsymbol{x} \in \boldsymbol{C}$.

Next, if $\boldsymbol{b} + \boldsymbol{y} \in \boldsymbol{b} + \boldsymbol{C}$, then,

$$\boldsymbol{b} + \boldsymbol{y} = (\boldsymbol{a} + \boldsymbol{x}) + \boldsymbol{y} = \boldsymbol{a} + (\boldsymbol{x} + \boldsymbol{y}) \in \boldsymbol{a} + \boldsymbol{C}.$$

Hence,

$\boldsymbol{b} + \boldsymbol{C} \subseteq \boldsymbol{a} + \boldsymbol{C}$. On the other hand, if $\boldsymbol{a} + \boldsymbol{z} \in \boldsymbol{a} + \boldsymbol{C}$, then,

$$\boldsymbol{a} + \boldsymbol{z} = (\boldsymbol{b} - \boldsymbol{x}) + \boldsymbol{z} = \boldsymbol{b} + (\boldsymbol{z} - \boldsymbol{x}) \in \boldsymbol{b} + \boldsymbol{C}.$$

Hence,

$$\boldsymbol{a} + \boldsymbol{C} \subseteq \boldsymbol{b} + \boldsymbol{C}, \text{ and so } \boldsymbol{b} + \boldsymbol{C} = \boldsymbol{a} + \boldsymbol{C}.$$

Definition 3.18 The vector having the minimum weight in a coset is called the **Coset Leader**. If there is more than one vector with the minimum weight, one of them is chosen at random and is declared the coset leader.

Example 3.17 Let $\boldsymbol{C}$ be the binary (3, 2) code with the generator matrix given by

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

i.e., $\boldsymbol{C} = \{000, 010, 101, 111\}$.

The cosets of $\boldsymbol{C}$ are,

$$000 + \boldsymbol{C} = 000, 010, 101, 111,$$
$$001 + \boldsymbol{C} = 001, 011, 100, 110.$$

Note that all the eight vectors have been covered by these two cosets. As we have already seen (in the above theorem), if $\boldsymbol{a} + \boldsymbol{C}$ is a coset of $\boldsymbol{C}$ and $\boldsymbol{b} \in \boldsymbol{a} + \boldsymbol{C}$, we have $\boldsymbol{b} + \boldsymbol{C} = \boldsymbol{a} + \boldsymbol{C}$.

Hence, all cosets have been listed. For the sake of illustration we write down the following

$$010 + \boldsymbol{C} = 010, 000, 111, 101,$$
$$011 + \boldsymbol{C} = 011, 001, 110, 101,$$
$$100 + \boldsymbol{C} = 100, 110, 001, 011,$$
$$101 + \boldsymbol{C} = 101, 111, 000, 101,$$
$$110 + \boldsymbol{C} = 110, 100, 011, 001,$$
$$111 + \boldsymbol{C} = 111, 101, 010, 000.$$

It can be seen that all these sets are already covered.

Since two cosets are either disjoint or coincide (from Theorem 3.6), the set of all vectors, $GF(q)^n$ can be written as

$$GF(q)^n = \boldsymbol{C} \cup (\boldsymbol{a}_1 + \boldsymbol{C}) \cup (\boldsymbol{a}_2 + \boldsymbol{C}) \cup \ldots \cup (\boldsymbol{a}_t + \boldsymbol{C})$$

where $t = q^{n-k} - 1$.

Definition 3.19 A **Standard Array** for an (n, k) code $\boldsymbol{C}$ is a $q^{n-k} \times q^k$ array of *all* vectors in $GF(q)^n$ in which the first row consists of the code $\boldsymbol{C}$ (with $\mathbf{0}$ on the extreme left), and the other rows are the cosets $\boldsymbol{a}_i + \boldsymbol{C}$, each arranged in corresponding order, with the coset leader on the left.

Steps for constructing a standard array:

(i) In the first row write down all the valid codewords, starting with the all-zero codeword.

(ii) Choose a vector $\boldsymbol{a}_1$ which is not in the first row. Write down the coset $\boldsymbol{a}_1 + \boldsymbol{C}$ as the second row such that $\boldsymbol{a}_1 + \boldsymbol{x}$ is written under $\boldsymbol{x} \in \boldsymbol{C}$.

(iii) Next choose another vector $\boldsymbol{a}_2$ (not present in the first two rows) of minimum weight and write down the coset $\boldsymbol{a}_2 + \boldsymbol{C}$ as the third row such that $\boldsymbol{a}_2 + \boldsymbol{x}$ is written under $\boldsymbol{x} \in \boldsymbol{C}$.

(iv) Continue the process until all the cosets are listed and every vector in $GF(q)^n$ appears exactly once.

Example 3.18 Consider the code $\boldsymbol{C} = \{0000, 1011, 0101, 1110\}$. The corresponding standard array is

codewords →	0000	1011	0101	1110
	1000	0011	1101	0110
	0100	1111	0001	1010
	0010	1001	0111	1100
	↑			
	coset leader			

Note that each entry is the sum of the codeword and its coset leader.

Let us now look at the concept of decoding (obtaining the information symbols from the received codewords) using the standard array. Since the standard array comprises all possible words belonging to $GF(q)^n$, the received word can always be identified with one of the elements of the standard array. If the received word is a valid codeword, it is concluded that no errors have occurred (this conclusion may be wrong with a very low probability of error, when one valid codeword gets modified to another valid codeword due to noise!). In the case that the received word, $\boldsymbol{v}$, does not belong to the set of valid codewords, we surmise that an error has occurred. The decoder then declares that the coset leader is the error vector, $\boldsymbol{e}$, and decodes the codeword as $\boldsymbol{v} - \boldsymbol{e}$. This is the codeword at the top of the column containing $\boldsymbol{v}$. Thus, mechanically, we decode the codeword as the one on the top of the column containing the received word.

Example 3.19 Suppose the code in the previous example $\boldsymbol{C} = \{0000, 1011, 0101, 1110\}$ is used and the received word is $\boldsymbol{v} = 1101$. Since it is not one of the valid codewords, we deduce that an error has occurred. Next we try to estimate which one of the four possible codewords was actually transmitted. If we make use of the standard array of the earlier example, we find that 1101 lies in the 3rd column. The topmost entry of this column is 0101. Hence the estimated codeword is 0101. Observe that:

$$d\,(1101, 0000) = 3,\ d\,(1101, 1011) = 2,$$
$$d\,(1101, 0101) = 1,\ d\,(1101, 1110) = 2$$

and the error vector $\boldsymbol{e} = 1000$, the coset leader.

Codes with larger blocklengths are desirable (though not always; see the concluding remarks on this chapter) because the code rates of larger codes perform closer to the **Shannon Limit.** As we go to larger codes (with larger values of k and n), the method of standard array will become less practical because the size of the standard array $(q^{n-k} \times q^k)$ will become unmanageably large. One of the basic objectives of coding theory is to develop efficient decoding strategies. If we are to build decoders that will work in real-time, the decoding scheme should be realizable both in terms of memory required as well as the computational load. Is it possible to reduce the standard array? The answer lies in the concept of Syndrome Decoding, which we are going to discuss next.

3.7 SYNDROME DECODING

The standard array can be simplified if we store only the first column, and compute the remaining columns, if needed. To do so, we introduce the concept of the ***Syndrome*** of the error pattern.

Definition 3.20 Suppose $\boldsymbol{H}$ is a parity check matrix of an (n, k) code, then for any vector $\boldsymbol{v} \in GF(q)^n$, the vector

$$\boldsymbol{s} = \boldsymbol{v}\boldsymbol{H}^T \tag{3.13}$$

is called the **Syndrome** of $\boldsymbol{v}$.

The syndrome of $\boldsymbol{v}$ is sometimes explicitly written as $\boldsymbol{s}(\boldsymbol{v})$. It is called a syndrome because it gives us the symptoms of the error, thereby helping us to diagnose the error.

Theorem 3.7 Two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are in the same coset of $\boldsymbol{C}$ if and only if they have the same syndrome.

Proof The vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ belong to the same coset

$$\Leftrightarrow x + C = y + C$$
$$\Leftrightarrow x - y \in C$$
$$\Leftrightarrow (x - y)H^T = 0$$
$$\Leftrightarrow xH^T = yH^T$$
$$\Leftrightarrow s(x) = s(y)$$

Thus, there is a one to one correspondence between cosets and syndromes.

We can reduce the size of the standard array by simply listing the syndromes and the corresponding coset leaders.

Example 3.20 We now extend the standard array listed in Example 3.18 by adding the syndrome column.

The code is C = {0000, 1011, 0101, 1110}. The corresponding standard array is

						Syndrome
Codewords	→	0000	1011	0101	1111	00
		1000	0011	1101	0110	11
		0100	1111	0001	1010	01
		0010	1001	0111	1100	10
		↑				
		coset leader				

The steps for syndrome decoding are as follows

(i) Determine the syndrome ($s = vH^T$) of the received word, v.
(ii) Locate the syndrome in the 'syndrome column'.
(iii) Determine the corresponding coset leader. This is the error vector, e.
(iv) Subtract this error vector from the received word to get the codeword $y = v - e$.

Having developed an efficient decoding methodology by means of syndrome decoding, let us now find out how much advantage coding actually provides.

3.8 ERROR PROBABILITY AFTER CODING (PROBABILITY OF ERROR CORRECTION)

Definition 3.21 The **Probability of Error** (or, the **Word Error Rate**) P_{err} for any decoding scheme is the probability that the decoder output is a wrong codeword. It is also called the **Residual Error Rate**.

Suppose there are M codewords (of length n) which are used with equal probability. Let the decoding be done using a standard array. Let the number of coset leaders with weight i be denoted by α_i. We assume that the channel is a BSC with symbol error probability p. A decoding error occurs if the error vector e is *not* a coset leader. Therefore, the probability of correct decoding will be

$$P_{cor} = \sum_{i=0}^{n} \alpha_i \, p^i \, (1-p)^{n-i} \tag{3.14}$$

Hence, the probability of error will be

$$P_{err} = 1 - \sum_{i=0}^{n} \alpha_i \, p^i \, (1-p)^{n-i} \tag{3.15}$$

Example 3.21 Consider the standard array in Example 3.18. The coset leaders are 0000, 1000, 0100 and 0010. Therefore $\alpha_0 = 1$ (only one coset leader with weight equal to zero), $\alpha_1 = 3$ (the remaining three are of weight one) and all other $\alpha_i = 0$.

Therefore,

$$P_{err} = 1 - [(1-p)^4 + 3p(1-p)^3]$$

Recall that this code has four codewords, and can be used to send 2 bits at a time. If we did not perform coding, the probability of error of the 2-bit message being received incorrectly would be

$$P_{err} = 1 - P_{cor} = 1 - (1-p)^2.$$

Note that for $p = 0.01$, the **Word Error Rate** (upon coding) is $P_{err} = 0.0103$, while for the uncoded case $P_{err} = 0.0199$. So, coding has almost halved the word error rate. The comparison of P_{err} for messages with and without coding is plotted in Fig. 3.3. It can be seen that coding outperforms the uncoded case only for $p < 0.5$. Note that the improvement due to coding comes at the cost of information transfer rate. In this example, the rate of information transfer has been cut down by half as we are sending two parity bits for every two information bits.

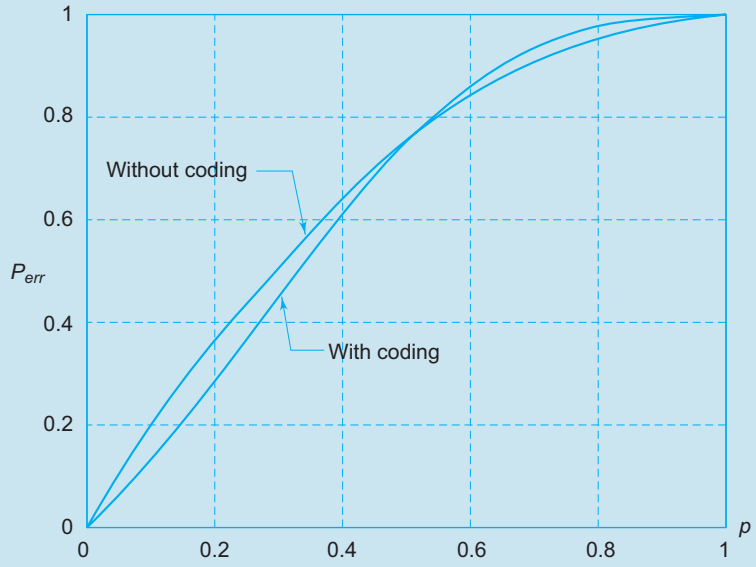


Fig. 3.3 *Comparison of P_{err} for Coded and Uncoded 2-Bit Messages.*

Example 3.22 This example will help us visualize the power of coding. Consider a BSC with the probability of symbol error $p = 10^{-7}$. Suppose 10 bit long words are being transmitted *without coding*. Let the bit rate of the transmitter be 10^7 b/s, which implies that 10^6 word/s are being sent. The probability that a word is received incorrectly is

$$\binom{10}{1}(1-p)^9 p + \binom{10}{2}(1-p)^8 p^2 + \binom{10}{3}(1-p)^7 p^3 + \ldots \approx \binom{10}{1}(1-p)^9 p = 10^{-6} \text{ word/s.}$$

Therefore, in one second, $10^{-6} \times 10^6 = 1$ word will be in error ! The implication is that every second a word will be in error and *it will not be detected.*

Next, let us add a parity bit to the uncoded words so as to make them 11 bits long. The parity makes all the codewords of even parity and thus ensures that a single bit in error will be detected. The only way that the coded word will be in error is if two or more bits get flipped, i.e., at least two bits are in error. This can be computed as 1 – probability that less than two bits are in error. Therefore, the probability of word error will be

$$1-(1-p)^{11} - \binom{11}{1}(1-p)^{10} p \approx 1-(1-11p) - 11(1-10p)\,p = 110\,p^2 = 11 \times 10^{-13}$$

The new word rate will be $10^7/11$ word/s because now 11 bits constitute one word and the bit rate is the same as before. Thus in one second, $(10^7/11) \times (11 \times 10^{-13}) = 10^{-6}$ words will be in error. This implies that, after coding, one word will be received incorrectly *without detection* every 10^6 seconds = 11.5 days!

So just by increasing the word length from 10 bits (uncoded) to 11 bits (with coding), we have been able to obtain a dramatic decrease in the Word Error Rate. For the second case, each time a word is detected to be in error, we can request the transmitter to re-transmit the word.

This strategy for retransmission is called the **A**utomatic **R**epeat Re**q**uest (ARQ).

3.9 PERFECT CODES

Definition 3.22 For any vector $\boldsymbol{u}$ in $GF(q)^n$ and any integer $r \geq 0$, the sphere of radius r and centre $\boldsymbol{u}$, denoted by $S(\boldsymbol{u}, r)$, is the set $\{\mathbf{v} \in GF(q)^n \mid d(\boldsymbol{u}, \boldsymbol{v}) \leq r\}$.

This definition can be interpreted graphically, as shown in Fig. 3.4. Consider a code $\boldsymbol{C}$ with minimum distance $d^*(\boldsymbol{C}) \geq 2t+1$. The spheres of radius t centred at the codewords $\{\boldsymbol{c}_1, \boldsymbol{c}_2, \ldots \boldsymbol{c}_M\}$ of $\boldsymbol{C}$ will then be disjoint. Now consider the decoding problem. Any received vector can be represented as a point in this space. If this point lies within a sphere, then by nearest neighbour decoding it will be decoded as the centre of the sphere. If t or fewer errors occur, the received word will definitely lie within the sphere of the codeword that was transmitted, and will be correctly decoded. If, however, larger than t errors occur, it will escape the sphere, thus resulting in incorrect decoding.

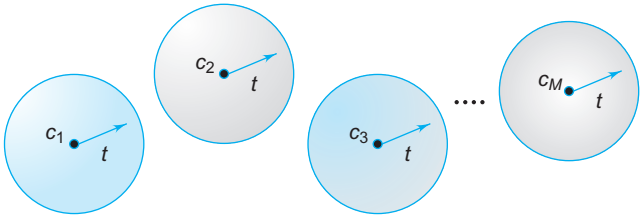


Fig. 3.4 *The concept of spheres in GF(q)^n.*

The codewords of the code with $d^*(\boldsymbol{C}) \geq 2t + 1$ are the centres of these non-overlapping spheres.

Theorem 3.8 A sphere of radius r $(0 \leq r \leq n)$ contains exactly

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r \text{ vectors.} \quad (3.16)$$

Proof Consider a vector $\boldsymbol{u}$ in $GF(q)^n$ and another vector $\boldsymbol{v}$ which is at a distance m from $\boldsymbol{u}$. This implies that the vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ differ at exactly m places. The total number of ways in which m position can be chosen from n positions is $\binom{n}{m}$. Now, each of these m places can be replaced by $(q-1)$ possible symbols. This is because the total size of the alphabet is q, out of which one is currently being used in that particular position in $\boldsymbol{u}$. Hence, the number of vectors at a distance exactly m from $\boldsymbol{u}$ is

$$\binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{r}(q-1)^r \quad (3.17)$$

Example 3.23 Consider a binary code (i.e., $q = 2$) and blocklength $n = 4$. The number of vectors at a distance 2 or less from any codeword will be

$$\binom{4}{0} + \binom{4}{1}(1) + \binom{4}{2}(1)^2 = 1 + 4 + 6 = 11$$

Without loss of generality we can choose the fixed vector $\boldsymbol{u} = 0000$. The vectors of distance 2 or less are

vectors at a distance 2: 0011, 1001, 1010, 1100, 0110, 0101,

vectors at a distance 1: 0001, 0010, 0100, 1000,

vectors at a distance 0: 0000.

Thus, there is a total of 11 such vectors.

Theorem 3.9 A q-ary (n, k) code with M codewords and minimum distance $(2t + 1)$ satisfies

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}\le q^n \qquad (3.18)$$

Proof Suppose $\boldsymbol{C}$ is a q-ary (n, k) code. Consider spheres of radius t centred on the M codewords. Each sphere of radius t has

$$\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t$$

vectors (theorem 3.8). Since none of the spheres intersect, the total number of vectors for the M disjoint spheres is $M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}$ which is upper bounded by q^n, the total number of vectors of length n in $GF(q)^n$.

This bound is called the **Hamming Bound** or the **Sphere Packing Bound** and it holds good for nonlinear codes as well. For binary codes, the **Hamming Bound** will become

$$M\left\{\binom{n}{0}+\binom{n}{1}+\binom{n}{2}+\cdots+\binom{n}{t}\right\}\le 2^n. \qquad (3.19)$$

It should be noted here that the mere existence of a set of integers n, M and t satisfying the Hamming bound, does not confirm it as a binary code. For example, the set $n = 5$, $M = 5$ and $t = 1$ satisfies the Hamming Bound. However, no binary code exists for this specification.

Observe that for the case when $M = q^k$, the Hamming Bound may be alternatively written as

$$\log_q\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}\le n-k. \qquad (3.20)$$

Definition 3.23 A **Perfect Code** is one which achieves the Hamming Bound, *i.e.*,

$$M\left\{\binom{n}{0}+\binom{n}{1}(q-1)+\binom{n}{2}(q-1)^2+\cdots+\binom{n}{t}(q-1)^t\right\}=q^n. \qquad (3.21)$$

For a Perfect Code, there are equal radius disjoint spheres centred at the codewords which completely fill the space. Thus, a t error correcting perfect code utilizes the entire space in the most efficient manner.

Example 3.24 Consider the Binary Repetition Code

$$C = \begin{cases} 00\ldots0 \\ 11\ldots1 \end{cases}$$

of block length n, where n is odd. In this case $M = 2$ and $t = (n-1)/2$. Upon substituting these values in the left hand side of the inequality for Hamming bound we get

$$\text{LHS} = \left\{ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{(n-1)/2} \right\} = 2.2^{n-1} = 2^n = \text{RHS}$$

Thus the repetition code is a Perfect Code. It is actually called a *Trivial* Perfect Code. In the next chapter, we shall see some examples of Non-trivial Perfect Codes.

One of the ways to search for perfect codes is to obtain the integer solutions for the parameters n, q, M and t in the equation for Hamming bound. Some of the solutions found by exhaustive computer search are listed below.

S.No.	n	q	M	t
1	23	2	2^{12}	3
2	90	2	2^{78}	2
3	11	3	3^6	2

3.10 HAMMING CODES

There are both binary and non-binary Hamming Codes. Here, we shall limit our discussion to binary Hamming Codes. The binary Hamming Codes have the property that

$$(n, k) = (2^m - 1, \quad 2^m - 1 - m) \tag{3.22}$$

where m is any positive integer. For example, for $m = 3$ we have a $(7, 4)$ Hamming Code. The parity check matrix, $\boldsymbol{H}$, of a Hamming Code is a very interesting matrix. Recall that the parity check matrix of an (n, k) code has $n - k$ rows and n columns. For the binary (n, k) Hamming code, the $n = 2^m - 1$ columns consist of all possible binary vectors with $n - k = m$ elements, except the all zero vector.

Example 3.25 The generator matrix for the binary (7, 4) Hamming Code is given by

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

The corresponding parity check matrix is

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

Observe that the columns of the parity check matrix consist of (100), (010), (101), (110), (111), (011) and (001). These seven are all the possible non-zero binary vectors of length three. It is quite easy to generate a systematic Hamming Code. The parity check matrix $\boldsymbol{H}$ can be arranged in the systematic form as follows

$$\boldsymbol{H} = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] = [-\boldsymbol{P}^{\boldsymbol{T}} \mid \boldsymbol{I}]\,.$$

Thus, the generator matrix in the systematic form for the binary Hamming code is

$$\boldsymbol{G} = [\boldsymbol{I} \mid \boldsymbol{P}] = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

From the above example, we observe that no two columns of $\boldsymbol{H}$ are linearly dependent (otherwise they would be identical). However, for $m > 1$, it is possible to identify three columns of $\boldsymbol{H}$ that would add up to zero. Thus, the minimum distance, d^*, of an (n, k) Hamming Code is equal to 3, which implies that it is a single-error correcting code. Hamming Codes are Perfect Codes.

By adding an overall parity bit, an (n, k) Hamming Code can be modified to yield an $(n+1, k)$ code with $d^* = 4$. On the other hand, an (n, k) Hamming Code can be shortened to an $(n - l, k - l)$ code by removing l rows of its generator matrix $\boldsymbol{G}$ or, equivalently, by removing l columns of its parity check matrix, $\boldsymbol{H}$. We can now give a more formal definition of Hamming Codes.

Definition 3.24 Let $n = (q^k - 1)/(q - 1)$. The (n, k) Hamming Code over $GF(q)$ is a code for which the parity check matrix has columns that are pairwise linearly independent (over $GF(q)$), i.e., the columns are a maximal set of pairwise linearly independent vectors.

3.11 OPTIMAL LINEAR CODES

Definition 3.25 For an (n, k, d^*) **Optimal Code**, no $(n-1, k, d^*)$, $(n+1, k+1, d^*)$ or $(n+1, k, d^*+1)$ code exists.

Optimal Linear Codes give the best distance property under the constraint of the block length. Most of the optimal codes have been found by long computer searches. It may be possible to have more than one optimal code for a given set of parameters n, k and d^*. For instance, there exist two different binary $(25, 5, 10)$ optimal codes.

Example 3.26 Consider the (24, 12, 8) binary code. It can be verified that

(i) a (23, 12, 8) code does not exist (only the (23, 12, 7) code exists),

(ii) a (25, 13, 8) code does not exist,

(iii) a (25, 12, 9) code does not exist.

Thus the binary (24, 12, 8) code is an optimal code.

3.12 MAXIMUM DISTANCE SEPARABLE (MDS) CODES

In this section we consider the problem of finding as large a minimum distance d^* possible for a given redundancy, r.

Theorem 3.10 An $(n, n-r, d^*)$ code satisfies $d^* \leq r + 1$.

Proof From the Singleton Bound we have $d^* \leq n - k + 1$.

Substitute $k = n - r$ to get $d^* \leq r + 1$.

Definition 3.26 An $(n, n-r, r+1)$ code is called a **Maximum Distance Separable** (MDS) code. An MDS code is a linear code of redundancy r, whose minimum distance is equal to $r + 1$.

3.13 CONCLUDING REMARKS

The classic paper by Claude Elwood Shannon in the *Bell System Technical Journal* in 1948 gave birth to two important fields (i) Information Theory and (ii) Coding Theory. At that time, Shannon was only 32 years old. According to Shannon's Channel Coding Theorem, *"the error rate of data transmitted over a band-limited noisy channel can be reduced to an arbitrarily small amount if*

the information rate is less than the channel capacity". Shannon predicted the existence of good channel codes but did not construct them. Since then the search for good codes has been on. Shannon's seminal paper can be accessed from the site:

http://cm.bell-labs.com/cm/ms/what/shannonday/paper.html.

In 1950, R.W. Hamming introduced the first single-error correcting code, which is still used today. The work on linear codes was extended by Golay (whose codes will be studied in the following chapter). Golay also introduced the concept of Perfect Codes. Non-binary Hamming Codes were developed by Golay and Cocke in the late 1950s. Lately, a lot of computer searches have been used to find interesting codes. However, some of the best known codes are ones discovered by sheer genius rather than exhaustive searches.

According to Shannon's theorem, if $C(p)$ represents the capacity (see Chapter 1 for further details) of a BSC with probability of bit error equal to p, then for arbitrarily low probability of symbol error we must have the code rate $R < C(p)$. Even though the channel capacity provides an upperbound on the achievable code rate $(R = k/n)$, evaluating a code exclusively against channel capacity may be misleading. The block length of the code, which translates directly into delay, is also an important parameter. Even if a code performs far from ideal, it is possible that it is the best possible code for a given rate and length. It has been observed that as we increase the block length of codes, the bounds on code rate are closer to channel capacity as opposed to codes with smaller blocklengths. However, longer blocklengths imply longer delays in decoding. This is because decoding of a codeword cannot begin until we have received the entire codeword. The maximum delay allowable is limited by practical constraints. For example, in mobile radio communications, packets of data are restricted to fewer than 200 bits. In these cases, codewords with very large blocklengths cannot be used.

SUMMARY

- A Word is a sequence of symbols. A Code is a set of vectors called codewords.
- The Hamming Weight of a codeword (or any vector) is equal to the number of non-zero elements in the codeword. The Hamming Weight of a codeword $\boldsymbol{c}$ is denoted by $w(\boldsymbol{c})$.
- A Block Code consists of a set of fixed length codewords. The fixed length of these codewords is called the Block Length and is typically denoted by n. A Block Coding Scheme converts a block of k information symbols to n coded symbols. Such a code is denoted by (n, k).
- The Code Rate of an (n, k) code is defined as the ratio (k/n), and reflects the fraction of the codeword that consists of the information symbols.
- The minimum distance of a code is the minimum Hamming Distance between any two codewords. An (n, k) code with minimum distance d^* is sometimes denoted by (n, k, d^*). The minimum weight of a code is the smallest weight of any non-zero codeword, and is

denoted by w^*. For a Linear Code the minimum distance is equal to the minimum weight of the code, i.e., $d^* = w^*$.

- A Linear Code has the following properties:
  (i) The sum of two codewords belonging to the code is also a codeword belonging to the code.
  (ii) The all-zero codeword is always a codeword.
  (iii) The minimum Hamming Distance between two codewords of a linear code is equal to the minimum weight of any non-zero codeword, i.e., $d^* = w^*$.
- The generator matrix converts (encodes) a vector of length k to a vector of length n. Let the input vector (uncoded symbols) be represented by $\boldsymbol{i}$. The coded symbols will be given by $\boldsymbol{c} = \boldsymbol{iG}$.
- Two q-ary codes are called equivalent if one can be obtained from the other by one or both operations listed below:
  (i) permutation of symbols appearing in a fixed position.
  (ii) permutation of position of the code.
- An (n, k) Systematic Code is one in which the first k symbols of the codeword of block length n are the information symbols themselves. A generator matrix of the form $\boldsymbol{G} = [\boldsymbol{I} \,|\, \boldsymbol{P}]$ is called the systematic form or the standard form of the generator matrix, where $\boldsymbol{I}$ is a $k \times k$ identity matrix and $\boldsymbol{P}$ is a $k \times (n - k)$ matrix.
- The Parity Check Matrix, $\boldsymbol{H}$, for the given code satisfies $\boldsymbol{cH}^T = \boldsymbol{0}$, where $\boldsymbol{c}$ is a valid codeword. Since $\boldsymbol{c} = \boldsymbol{iG}$, therefore, $\boldsymbol{iGH}^T = \boldsymbol{0}$. The Parity Check Matrix is not unique for a given code.
- A Maximum Distance Code satisfies $d^* = n - k + 1$.
- For a code to be able to correct up to t errors, we must have $d^* \geq 2t + 1$, where d^* is minimum distance of the code.
- Let $\boldsymbol{C}$ be an (n, k) code over $GF(q)$ and $\boldsymbol{a}$ be any vector of length n. Then the set $\boldsymbol{a} + \boldsymbol{C} = \{\boldsymbol{a} + \boldsymbol{x} \mid \boldsymbol{x} \in \boldsymbol{C}\}$ is called a coset (or translate) of $\boldsymbol{C}$. $\boldsymbol{a}$ and $\boldsymbol{b}$ are said to be in the same coset iff $(\boldsymbol{a} - \boldsymbol{b}) \in \boldsymbol{C}$.
- Suppose $\boldsymbol{H}$ is a Parity Check Matrix of an (n, k) code. Then for any vector $\boldsymbol{v} \in GF(q)^n$, the vector $\boldsymbol{s} = \boldsymbol{vH}^T$ is called the Syndrome of $\boldsymbol{v}$. It is called a syndrome because it gives us the symptoms of the error, thereby helping us to diagnose the error.
- A Perfect Code achieves the Hamming Bound, i.e.,

$$M = \left\{ \binom{n}{0} + \binom{n}{1}(q-1) + \binom{n}{2}(q-1)^2 + \cdots + \binom{n}{t}(q-1)^t \right\} = q^n$$

- The binary Hamming Codes have the property that $(n, k) = (2^m - 1, 2^m - 1 - m)$, where m is any positive integer. Hamming Codes are Perfect Codes.
- For an (n, k, d^*) Optimal Code, no $(n - 1, k, d^*)$, $(n + 1, k + 1, d^*)$ or $(n + 1, k, d^* + 1)$ code exists.
- An $(n, n - r, r + 1)$ code is called a Maximum Distance Separable (MDS) Code. An MDS code is a linear code of redundancy r, whose minimum distance is equal to $r + 1$.

The greatest unsolved theorem in mathematics is why some people are better at it than others.

Mathesis, Adrian

PROBLEMS

3.1 Show that $\boldsymbol{C} = \{0000, 1100, 0011, 1111\}$ is a linear code. What is its minimum distance?

3.2 Construct, if possible, binary (n, k, d^*) codes with the following parameters:

(i) (6, 1, 6)
(ii) (3, 3, 1)
(iii) (4, 3, 2)

3.3 Consider the following generator matrix over $GF(2)$

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

(i) Generate all possible codewords using this matrix.
(ii) Find the parity check matrix, $\boldsymbol{H}$.
(iii) Find the generator matrix of an equivalent systematic code.
(iv) Construct the standard array for this code.
(v) What is the minimum distance of this code?
(vi) How many errors can this code detect?
(vii) Write down the set of error patterns this code can detect.
(viii) How many errors can this code correct?
(ix) What is the probability of symbol error if we use this encoding scheme? Compare it with the uncoded probability of error.
(x) Is this a linear code?

3.4 For the code $\boldsymbol{C} = \{00000, 10101, 01010, 11111\}$ construct the generator matrix. Since this $\boldsymbol{G}$ is not unique, suggest another generator matrix that can also generate this set of codewords.

3.5 Show that if there is a binary (n, k, d) code with d^* even, then there exists a binary (n, k, d^*) code in which all codewords have even weight.

3.6 Show that if $\boldsymbol{C}$ is a binary linear code, then the code obtained by adding an overall parity check bit to $\boldsymbol{C}$ is also linear.

3.7 For each of the following sets S, list the code $<S>$.

(a) $\boldsymbol{S} = \{0101, 1010, 1100\}$.
(b) $\boldsymbol{S} = \{1000, 0100, 0010, 0001\}$.
(c) $\boldsymbol{S} = \{11000, 01111, 11110, 01010\}$.

3.8 Consider the (23, 12, 7) binary code. Show that if it is used over a binary symmetric channel (BSC) with probability of bit error $p = 0.01$, the word error will be approximately 0.00008.

3.9 Suppose $\boldsymbol{C}$ is a binary code with parity check matrix, $\boldsymbol{H}$. Show that the extended code $\boldsymbol{C}_1$, obtained from $\boldsymbol{C}$ by adding an overall parity bit, has the parity check matrix

$$\boldsymbol{H}_1 = \left[\begin{array}{cccccc} & & & & | & 0 \\ & & & & | & 0 \\ & H & & & | & \\ & & & & | & 0 \\ - & - & - & - & - & - \\ 1 & 1 & \cdots & 1 & | & 1 \end{array}\right]$$

3.10 For a (5, 3) code over $GF(4)$, the generator matrix is given by

$$\boldsymbol{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 2 \\ 0 & 0 & 1 & 1 & 3 \end{bmatrix}$$

(i) Find the parity check matrix.
(ii) How many errors can this code detect?
(iii) How many errors can this code correct?
(iv) How many erasures can this code correct?
(v) Is this a perfect code?

3.11 Let $\boldsymbol{C}$ be a binary perfect code of length n with minimum distance 7. Show that $n = 7$ or $n = 23$.

3.12 Let r_H denote the code rate for the binary Hamming code. Determine $\lim_{k \to \infty} r_H$.

3.13 Show that a (15, 8, 5) code does not exist.

COMPUTER PROBLEMS

3.14 Write a computer program to find the minimum distance of a Linear Block Code over $GF(2)$, given the generator matrix for the code.

3.15 Generalize the above program to find the minimum distance of any Linear Block Code over $GF(q)$.

3.16 Write a computer program to exhaustively search for all the perfect code parameters n, q, M and t in the equation for the Hamming Bound. Search for $1 \geq n \geq 200$, $2 \geq q \geq 11$.

3.17 Write a computer program for a universal binary Hamming encoder with rate $\dfrac{2^m - 1}{2^m - 1 - m}$.

The program should take as input the value of m and a bit-stream to be encoded. It should then generate an encoded bit-stream. Develop a program for the decoder also.

Now, perform the following tasks:

(i) Write an error generator module that takes in a bit stream and outputs another bit-stream after inverting every bit with probability p, i.e., the probability of a bit error is p.

(ii) For $m = 3$, pass the Hamming encoded bit-stream through the above-mentioned module and then decode the received words using the decoder block.

(iii) Plot the residual error probability (the probability of error after decoding) as a function of p. Note that if you are working in the range of BER $= 10^{-r}$, you must transmit of the order of 10^{r+2} bits (why?).

(iv) Repeat your simulations for $m = 5$, 8 and 15. What happens as $m \to \infty$.