Chapter 6

Error-Correcting Codes

6.1 Introduction

We have seen that the existence of redundancy makes it possible to compress data and so reduce the amount of space or time taken to transmit or store it. A consequence of removing redundancy is that the data become more susceptible to noise.

Conversely, it is possible to increase the redundancy of a set of data by adding to it. The added redundancy can be used to detect when the data have been corrupted by noise and even to undo the corruption. This idea is the the basis of the theory of *error-correcting codes*.

EXAMPLE 6.1

The simplest example of an error-correcting code is the *parity check*. (A special case of this was discussed in Example 5.5 of Chapter 5.) Suppose that we have a sequence of binary digits that is to be transmitted over a noisy channel that may have the effect of changing some of the 1's to 0's and some of the 0's to 1's. We can implement a parity check by breaking the sequence into blocks of, say, seven bits and adding a redundant bit to make a block of eight bits.

The added bit is a parity bit. It is 0 if there is an even number of 1s in the seven-bit block and 1 if there is an odd number of 1s in the seven-bit block. Alternatively, it is the result of ANDing all the seven bits together.

When the eight bits are transmitted over a noisy channel, there may be one or more differences between the block that was transmitted and the block that is received. Suppose, first that the parity bit is different. It will then not be consistent with the other seven bits. If the parity bit is unchanged, but one of the other seven bits is changed, the parity bit will once again be inconsistent with the other seven bits. Checking to see if the parity bit is consistent with the other seven bits enables us to check whether there has been a single error in the transmission of the eight bit block. It does not tell us where the error has occurred, so we cannot correct it.

The parity check also cannot tell us whether more than one error has occurred. If two bits are changed, the parity bit will still be consistent with the other seven bits. The

same is true if four or six bits are changed. If three bits are changed, the parity bit will not be consistent with the other seven bits, but this is no different from the situation where only one bit is changed. The same is true if five or seven bits are changed.

The parity check can be carried out easily by ANDing all eight bits together. If the result is 0, then either no error has occurred, or two, four or six errors have occurred. If the result is 1, then one, three, five or seven errors have occurred.

The parity check is a simple error-detection technique with many limitations. More effective error-correcting capabilities require more sophisticated approaches. □

EXAMPLE 6.2

Another simple way of introducing redundancy (discussed in Chapter 5, Example 5.6) is by repetition. If we have a sequence of bits, we can send each bit three times, so that the sequence 01100011 is transmitted as 000111111000000000111111. At the receiving end, the sequence is broken up into blocks of three bits. If there is an error in that block, one of the bits will be different, for example 001 instead of 000. In this case, we take the most frequently occurring bit as the correct one. This scheme allows us to detect and correct one error. If two errors occur within a block, they will be detected, but the correction procedure will give the wrong answer as the majority of bits will be erroneous. If all three bits are changed, this will not be detected.

If it is likely that errors will occur in bursts, this scheme can be modified by breaking the original sequence into blocks and transmitting the blocks three times each. The error correction procedure will then be applied to the corresponding bits in each of the three copies of the block at the receiving end.

This is an inefficient way of adding redundancy, as it triples the amount of data but only makes it possible to correct single errors. □

The theory of error-correcting codes uses many concepts and results from the mathematical subject of *abstract algebra*. In particular, it uses the notions of *rings*, *fields* and *linear spaces*. The following sections give an introduction to these notions. A rigorous treatment of this material can be found in any textbook on abstract algebra, such as [3].

6.2 Groups

A group is a collection of objects that can be combined in pairs to produce another object of the collection according to certain rules that require that there be an object

that does not change anything it combines with, and objects that undo the changes resulting from combining with other objects. The formal definition is as follows.

DEFINITION 6.1 Group *A* group, $(G, *)$, *is a pair consisting of a set* G *and an operation* $*$ *on that set, that is, a function from the Cartesian product* $G \times G$ *to* G*, with the result of operating on* a *and* b *denoted by* $a * b$*, which satisfies*

1. associativity: $a * (b * c) = (a * b) * c$ *for all* $a, b, c \in G$*;*
2. existence of identity: *there exists* $e \in G$ *such that* $e * a = a$ *and* $a * e = a$ *for all* $a \in G$*;*
3. existence of inverses: *for each* $a \in G$ *there exists* $a^{-1} \in G$ *such that* $a * a^{-1} = e$ *and* $a^{-1} * a = e$.

If the operation also satisfies the condition that $a * b = b * a$ for all $a, b \in G$ (*commutativity*), the group is called a *commutative group* or an *Abelian group*. It is common to denote the operation of an Abelian group by $+$, its identity element by 0 and the inverse of $a \in G$ by $-a$.

EXAMPLE 6.3

The simplest group has just two elements. It is Abelian, so we can denote the elements by 0 and g and the operation by $+$. From the definition, we must have $0 + 0 = 0$, $0 + g = g$, and $g + 0 = g$. g must be its own inverse, so $g + g = 0$. We can describe the operation by the following table where the first operand is listed in the column on the left, the second operand is listed in the row at the top and the results of the operation appear in the body of the table.

$+$	0	g
0	0	g
g	g	0

This table will be symmetric about the main diagonal if and only if the group is Abelian. □

When we use an operation table to define a group operation, each element of the group will appear exactly once in each row and each column of the body of the table.

DEFINITION 6.2 Order of a Group *The* order of a group *is the number of elements in it.*

EXAMPLE 6.4

It is easy to show that there is only one group of order 3. Let us denote the elements of the group by $0, 1, 2$ and the operation by $+$.

If we let 0 denote the identity element, we can start building the operation table as follows:

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

Now suppose that $1 + 2 = 1$. This would make 1 appear twice in the second row of the table. This breaks the rule that each element appears exactly once in each row of the body of the table, so we cannot have $1+2 = 1$. If $1+2 = 2$, 2 would appear twice in the third column of the table, so we cannot have this either. This leaves $1 + 2 = 0$, which does not break any rules. To complete the second row of the table we have to have $1 + 1 = 2$, and the table now looks like this:

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

We can now fill in the second and third columns with the missing elements to complete the table:

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

This is the operation table for addition modulo 3. □

In the example above, it is claimed that there is "only one group of order 3." This is true in the sense that any set of three elements with an operation that satisfies the group axioms must have the same structure as the group

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

regardless of the names of the elements. We regard

$+$	α	β	γ
α	α	β	γ
β	β	γ	α
γ	γ	α	β

as the same group because, if we replace α by 0, β by 1 and γ by 2 everywhere in the table above, we will finish up with the previous table.

To make these ideas precise, we need to consider functions that preserve the structure of the group in the following way.

DEFINITION 6.3 ***Group Homomorphism*** *Let $(G, *)$ and $(H, \circ)$ be two groups and let $h : G \to H$ be a function defined on G with values in H. h is a* group homomorphism *if it preserves the structure of G, that is, if*

$$h(a * b) = h(a) \circ h(b) \tag{6.1}$$

for all $a, b \in G$.

DEFINITION 6.4 ***Group Isomorphism*** *A group homomorphism that is both one-one and onto is a* group isomorphism.

Two groups that are isomorphic have the same structure and the same number of elements.

If $h : G \to H$ is a homomorphism, the image of the identity of G is the identity of H.

EXAMPLE 6.5

There are two groups of order 4. We will denote the elements of the first by $0, 1, 2, 3$ and use $+$ for the operation. The operation table is:

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

For the other, we will use e, a, b, c for the elements and $*$ for the operation. The operation table is

$*$	e	a	b	c
e	e	a	b	c
a	a	e	c	b
b	b	c	e	a
c	c	b	a	e

These groups obviously do not have the same structure. In the first group $1 + 1 = 2$ and $3 + 3 = 2$, but in the second group combining any element with itself under $*$ gives the identity, e. □

EXAMPLE 6.6

There is one group of order 5. Its operation table is

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

□

EXAMPLE 6.7

There are two groups of order 6. The operation table of the first is

$+$	0	1	2	3	4	5
0	0	1	2	3	4	5
1	1	2	3	4	5	0
2	2	3	4	5	0	1
3	3	4	5	0	1	2
4	4	5	0	1	2	3
5	5	0	1	2	3	4

For the second, we will use E, F, G, H, I, J for the elements and $*$ for the operation. The operation table is

$*$	E	F	G	H	I	J
E	E	F	G	H	I	J
F	F	E	I	J	G	H
G	G	H	E	F	J	I
H	H	G	J	I	E	F
I	I	J	F	E	H	G
J	J	I	H	G	F	E

The table is not symmetric, so this group is not Abelian. This is the smallest example of a non-commutative group. ∎

For any positive integer p, there is at least one group of order p. There may be more than one group of order p.

DEFINITION 6.5 The Cyclic Groups *For each positive integer p, there is a group called the* cyclic group of order *p, with set of elements*

$$\mathbb{Z}_p = \{0, 1, \ldots, (p-1)\}$$

and operation $\oplus$ defined by

$$i \oplus j = i + j \tag{6.2}$$

if $i + j < p$, where $+$ denotes the usual operation of addition of integers, and

$$i \oplus j = i + j - p \tag{6.3}$$

if $i + j > p$, where $-$ denotes the usual operation of subtraction of integers. The operation in the cyclic group is addition modulo *p. We shall use the sign $+$ instead of $\oplus$ to denote this operation in what follows and refer to "the cyclic group $(\mathbb{Z}_p, +)$," or simply "the cyclic group $\mathbb{Z}_p$."*

There are also plenty of examples of infinite groups.

EXAMPLE 6.8

The set of integers, $\mathbb{Z} = \{\ldots -3, -2, -1, 0, 1, 2, 3, \ldots\}$, with the operation of addition is a group. 0 is the identity and $-k$ is the inverse of k. This is an Abelian group. ∎

EXAMPLE 6.9

The set of real numbers $\mathbb{R}$ with the operation of addition is another Abelian group. Again, 0 is the identity and $-r$ is the inverse of $r \in \mathbb{R}$. ∎

EXAMPLE 6.10

The set of positive real numbers $\mathbb{R}^+$ with the operation of multiplication is also an Abelian group. This time, the identity is 1 and the inverse of $r \in \mathbb{R}^+$ is $1/r$. □

EXAMPLE 6.11

The set of $n \times n$ real matrices with the operation of matrix addition is a group. The zero matrix is the identity and the inverse of a matrix M is the matrix $-M$, consisting of the same elements as M but with their signs reversed. □

EXAMPLE 6.12

The set of non-singular $n \times n$ real matrices with the operation of matrix multiplication is a group for any positive integer n. The identity matrix is the group identity and the inverse of a matrix is its group inverse. These groups are not commutative. □

Groups are not used directly in the construction of error-correcting codes. They form the basis for more complex algebraic structures which are used.

6.3 Rings and Fields

Having two operations that interact with each other makes things more interesting.

DEFINITION 6.6 Ring *A* ring *is a triple* $(R, +, \times)$ *consisting of a set* R*, and two operations* $+$ *and* $\times$*, referred to as* addition *and* multiplication*, respectively, which satisfy the following conditions:*

1. associativity of $+$: $a + (b + c) = (a + b) + c$ *for all* $a, b, c \in R$*;*
2. commutativity of $+$: $a + b = b + a$ *for all* $a, b \in R$*;*
3. existence of additive identity: *there exists* $0 \in R$ *such that* $0 + a = a$ *and* $a + 0 = a$ *for all* $a \in R$*;*
4. existence of additive inverses: *for each* $a \in R$ *there exists* $-a \in R$ *such that* $a + (-a) = 0$ *and* $(-a) + a = 0$*;*
5. associativity of $\times$: $a \times (b \times c) = (a \times b) \times c$ *for all* $a, b, c \in R$*;*
6. distributivity of $\times$ over $+$: $a \times (b + c) = (a \times b) + (a \times c)$ *for all* $a, b, c \in R$.

The additive part of a ring, $(R, +)$, is an Abelian group.

In any ring $(R, +, \times)$, $0 \times r = 0$ and $r \times 0 = 0$ for all $r \in R$.

In many cases, it is useful to consider rings with additional properties.

DEFINITION 6.7 Commutative Ring *A ring $(R, +, \times)$ is a* commutative ring *if $\times$ is a commutative operation, that is, $a \times b = b \times a$ for all $a, b \in R$.*

DEFINITION 6.8 Ring with Unity *A ring $(R, +, \times)$ is a* ring with unity *if it has an identity element of the multiplication operation, that is, if there exists $1 \in R$ such that $1 \times a = a$ and $a \times 1 = a$ for all $a \in R$.*

DEFINITION 6.9 Division Ring *A ring $(R, +, \times)$ is a* division ring *if it is a ring with unity and every non-zero element has a multiplicative inverse, that is, for every $a \in R$, $a \neq 0$, there exists $a^{-1} \in R$ such that $a \times a^{-1} = 1$ and $a^{-1} \times a = 1$.*

DEFINITION 6.10 Field *A commutative division ring in which $0 \neq 1$ is a* field.

EXAMPLE 6.13

There is one ring with two elements. The additive part of the ring is the cyclic group of order 2, with operation table

$+$	0	1
0	0	1
1	1	0

and the multiplicative part has the operation table

$\times$	0	1
0	0	0
1	0	1

These tables represent the arithmetic operations of addition and multiplication modulo 2. The ring will be denoted by $(\mathbb{Z}_2, +, \times)$ or simply by $\mathbb{Z}_2$.

If we consider 0 and 1 to represent truth values, then these tables are the truth tables of the bit operations AND and XOR, respectively.

$\mathbb{Z}_2$ is a commutative ring, a ring with unity and a division ring, and hence a field. It is the smallest field and the one of most relevance to the construction of error-correcting codes. □

We can add a multiplicative structure to the cyclic groups to form rings.

DEFINITION 6.11 The Cyclic Rings *For every positive integer p, there is a ring $(\mathbb{Z}_p, +, \times)$, called the* cyclic ring of order *p, with set of elements*

$$Z_p = \{0, 1, \ldots, (p-1)\}$$

and operations $+$ denoting addition modulo p, and $\times$ denoting multiplication modulo p.

The previous example described the cyclic ring of order 2. The following examples show the operation tables of larger cyclic rings.

EXAMPLE 6.14

The operation tables of the cyclic ring of order 3 are

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

$\times$	0	1	2
0	0	0	0
1	0	1	2
2	0	2	1

$\mathbb{Z}_3$ is a field. □

EXAMPLE 6.15

The operation tables of the cyclic ring of order 4 are

$+$	0	1	2	3
0	0	1	2	3
1	1	2	3	0
2	2	3	0	1
3	3	0	1	2

$\times$	0	1	2	3
0	0	0	0	0
1	0	1	2	3
2	0	2	0	2
3	0	3	2	1

$\mathbb{Z}_4$ is not a field; 2 does not have a multiplicative inverse. □

EXAMPLE 6.16

The operation tables of the cyclic ring of order 5 are

$+$	0	1	2	3	4
0	0	1	2	3	4
1	1	2	3	4	0
2	2	3	4	0	1
3	3	4	0	1	2
4	4	0	1	2	3

$\times$	0	1	2	3	4
0	0	0	0	0	0
1	0	1	2	3	4
2	0	2	4	1	3
3	0	3	1	4	2
4	0	4	3	2	1

$\mathbb{Z}_5$ is a field. □

The cyclic ring of order p is a field if and only if p is a prime number.

The following are examples of infinite rings.

EXAMPLE 6.17

The integers $(\mathbb{Z}, +, \times)$ with the usual operations of addition and multiplication form a commutative ring with unity. It is not a division ring. □

EXAMPLE 6.18

The real numbers $(\mathbb{R}, +, \times)$ with the usual operations of addition and multiplication form a field. □

EXAMPLE 6.19

The complex numbers $(\mathbb{C}, +, \times)$ with the operations of complex addition and complex multiplication form a field. □

EXAMPLE 6.20

The set of $n \times n$ real matrices with operations matrix operation and matrix multiplication forms a ring with unity. The zero matrix is the additive identity and the identity matrix is the multiplicative identity. It is not commutative, and is not a division ring as only non-singular matrices have multiplicative inverses. □

Just as in the case of groups, we can talk about rings having the same structure and about functions which preserve structure.

DEFINITION 6.12 Ring Homomorphism *Let $(R, +, \times)$ and $(S, \oplus, \otimes)$ be two rings and let $h : R \to S$ be a function defined on R with values in S. h is a* ring homomorphism *if it preserves the structure of R, that is, if*

$$h(a + b) = h(a) \oplus h(b) \tag{6.4}$$

and

$$h(a \times b) = h(a) \otimes h(b) \tag{6.5}$$

for all $a, b \in G$.

DEFINITION 6.13 Ring Isomorphism *A ring homomorphism that is both one-one and onto is a* ring isomorphism.

As with groups, two rings are isomorphic if they have the same structure and the same number of elements.

6.4 Linear Spaces

Linear spaces consist of things which can be added, subtracted and re-scaled.

DEFINITION 6.14 Linear Space *A* linear space *over a field F is a 6-tuple $(V, F, \oplus, +, \times, \circ)$, where $(V, \oplus)$ is a commutative group, $(F, +, \times)$ is a field and $\circ : F \times V \to V$ is a function that satisfies the following conditions:*

1. *$a \circ (b \circ v) = (a \times b) \circ v$ for all $a, b \in F$ and all $v \in V$;*
2. *$(a + b) \circ v = (a \circ v) \oplus (b \circ v)$ for all $a, b \in F$ and all $v \in V$;*
3. *$a \circ (v \oplus w) = (a \circ v) \oplus (a \circ w)$ for all $a \in F$ and all $v, w \in V$;*
4. *$1 \circ v = v$ for all $v \in V$.*

Most of the elementary examples of linear spaces are spaces of geometric vectors; so an alternative name for a linear space is a *vector space*. The elements of V are called *vectors* and the operation $\oplus$ is called *vector addition*, while the elements of f are called scalars and the function $\circ : F \times V \to V$ is called *multiplication by scalars*.

While we have been careful to use different symbols for the various operations in the definition above, we shall hereafter abuse notation by using the same symbol, $+$, for addition in V as well as for addition in F, and by omitting the symbol $\circ$ and denoting

multiplication by a scalar by juxtaposition of a scalar and a vector. We shall also use the same symbol, 0, for both the group identity in V and the additive identity in F.

The following properties are simple consequences of the definition above:

1. $0v = 0$ for all $v \in V$, where the 0 on the left hand side belongs to F and the 0 on the right hand side belongs to V;
2. $a0 = 0$ for all $a \in F$, where the 0 on both sides belongs to V;
3. $(-a)v = a(-v) = -(av)$ for all $a \in F$ and $v \in V$.

The archetypal linear spaces are the n-dimensional real vector spaces.

EXAMPLE 6.21

Let $\mathbb{R}^n$ denote the n-fold Cartesian product $\mathbb{R} \times \mathbb{R} \times \ldots \times \mathbb{R}$, consisting of n-tuples $(x^1, x^2, \ldots, x^n)$ for $x^i \in \mathbb{R}$, $1 \leq i \leq n$. If we define vector addition by

$$(x^1, x^2, \ldots, x^n) + (y^1, y^2, \ldots, y^n) = (x^1 + y^1, x^2 + y^2, \ldots, x^n + y^n)$$

and multiplication by a scalar by

$$a(x^1, x^2, \ldots, x^n) = (ax^1, ax^2, \ldots, ax^n)$$

for $(x^1, x^2, \ldots, x^n)$ and $(y^1, y^2, \ldots, y^n) \in \mathbb{R}^n$ and $a \in \mathbb{R}$, $\mathbb{R}^n$ becomes a linear space over $\mathbb{R}$. ∎

EXAMPLE 6.22

For an example of a linear space that is not $\mathbb{R}^n$, consider the set $\mathcal{C}$ of continuous functions on the real line. For f and $g \in \mathcal{C}$, define the sum $f + g$ by

$$(f + g)(x) = f(x) + g(x),$$

for all $x \in \mathbb{R}$, and the product of $r \in \mathbb{R}$ and $f \in \mathcal{C}$ by

$$(rf)(x) = rf(x),$$

for all $x \in \mathbb{R}$, where the right hand side of the equation denotes the product of r and $f(x)$ as real numbers.

If f and g are continuous, so are $f + g$ and rf. The other properties that addition in $\mathcal{C}$ and multiplication by scalars must satisfy follow from the properties of the real numbers. ∎

Parts of linear spaces can be linear spaces in their own right.

DEFINITION 6.15 Linear Subspace *A subset S of a linear space V over a field F is a* linear subspace *of V if for all $v, w \in S$ and all $a \in F$, $v + w \in S$ and $av \in S$.*

EXAMPLE 6.23

If $x = (x^1, x^2, \ldots, x^n) \in \mathbb{R}^n$ then the set

$$S_1 = \{rx : r \in \mathbb{R}\}$$

is a linear subspace of $\mathbb{R}^n$. Geometrically, this is a line through the origin.

If $y = (y^1, y^2, \ldots, y^n)$ also belongs to $\mathbb{R}^n$ then the set

$$S_2 = \{rx + sy : r, s \in \mathbb{R}\}$$

is a linear subspace of $\mathbb{R}^n$. This is a plane through the origin. □

EXAMPLE 6.24

The set $\{f \in \mathcal{C} : f(0) = 0\}$ is a linear subspace of $\mathcal{C}$. To check this, note that if $f(0) = 0$ and $g(0) = 0$, then $(f + g)(0) = f(0) + g(0) = 0$ and if $a \in \mathbb{R}$, then $(rf(0)) = rf(0) = 0$. □

It is quite easy to create subspaces of a linear space.

DEFINITION 6.16 Linear Combination *If V is a linear space over F, a* (finite) linear combination *of elements of V is a sum*

$$c = \sum_{i=1}^{k} a_i s_i \tag{6.6}$$

where k is a positive integer, the $a_i \in F$ and the $s_i \in V$.

This definition includes the case where $k = 1$.

DEFINITION 6.17 Linear Span *Let S be a subset of the linear space V. The* linear span *of S is the set*

$$span[S] = \left\{ \sum_{i=1}^{k} a_i s_i : k \geq 1, a_i \in F, s_i \in S \right\} \tag{6.7}$$

consisting of all finite linear combinations of elements of S.

The linear span of S is also known as the *linear subspace generated by* S.

It is possible for different sets to generate the same linear subspace.

EXAMPLE 6.25

Consider the linear subspace of $\mathbb{R}^3$ generated by the vectors $(1,0,0)$ and $(0,1,0)$. This is the set $\{(x,y,0) \in \mathbb{R}^3, x,y \in \mathbb{R}\}$. It is also generated by the vectors $(1,1,0)$, and $(1,-1,0)$.

The same subspace is also generated by the vectors $(1,0,0)$, $(0,1,0)$ and $(3,-2,0)$. There is a redundancy in this, since we can express $(3,-2,0)$ as a linear combination of the other two vectors,

$$(3,-2,0) = 3(1,0,0) + (-2)(0,1,0),$$

and so we can reduce any linear combination of the three vectors to a linear combination of the first two by putting

$$a(1,0,0) + b(0,1,0) + c(3,-2,0) = (a+3c)(1,0,0) + (b-2c)(0,1,0),$$

for any $a, b, c \in \mathbb{R}$. ∎

It is important to distinguish sets that have the redundancy property illustrated in the example above from sets that do not possess this property.

DEFINITION 6.18 Linearly Independent *A subset S of a linear space V over F is* linearly independent *if for any set of vectors $\{s_1, s_2, \ldots, s_n\}$ contained in S, the equation*

$$\sum_{i=1}^{n} a_i s_i = 0 \tag{6.8}$$

implies that all the $a_i = 0$.

A set that is not linearly independent is *linearly dependent*.

EXAMPLE 6.26

$\{(1,1,1),(1,-1,1)\}$ is a linearly independent subset of $\mathbb{R}^3$. The equation

$$a_1(1,1,1) + a_2(1,-1,1) = (0,0,0)$$

implies the three equations

$$\begin{aligned} a_1 + a_2 &= 0 \\ a_1 - a_2 &= 0 \\ a_1 + a_2 &= 0. \end{aligned}$$

The first and third equation are identical, but the only solution of the first and second equations is $a_1 = 0, a_2 = 0$.

$\{(1,1,1),(1,-1,1),(0,1,0)\}$ is a linearly dependent subset of $\mathbb{R}^3$. The equation

$$a_1(1,1,1) + a_2(1,-1,1) + a_3(0,-1,0) = (0,0,0)$$

implies the three equations

$$\begin{aligned} a_1 + a_2 &= 0 \\ a_1 - a_2 - a_3 &= 0 \\ a_1 + a_2 &= 0. \end{aligned}$$

This set of equations has infinitely many solutions, for example, $a_1 = 1$, $a_2 = -1$, $a_3 = 2$. □

A linearly independent set that generates a vector space has the property that removing any vector from the set will produce a set that no longer generates the space while adding a vector to the set will produce a set that is no longer linearly independent. Such sets are important enough to be given a special name.

DEFINITION 6.19 Basis *If V is a linear space over F, a* basis *for V is a linearly independent subset of V that generates the whole of V.*

EXAMPLE 6.27

The set of vectors $\{(1,0,0,\ldots,0),(0,1,0,\ldots,0),(0,0,1,\ldots,0),\ldots,(0,0,0,\ldots,1)\}$, forms a basis for $\mathbb{R}^n$, known as the standard basis. There are lots of others.

For $n = 3$, the standard basis is $\{(1,0,0),(0,1,0),(0,0,1)\}$. Other bases are $\{(1,1,0),(1,-1,0),(0,0,-1)\}$, and $\{(1,-2,3),(1,2,1),(4,-1,-2)\}$. □

It can be shown that every basis of a linear space contains the same number of vectors. This makes the following definition unambiguous.

DEFINITION 6.20 Dimension *The* dimension *of a linear space is the number of elements in any basis for the space.*

EXAMPLE 6.28

The existence of the standard basis for $\mathbb{R}^n$ shows that its dimension is n. □

The dimension of a linear space is also the maximum number of elements that can be contained in a linearly independent subset of that space. There are linear spaces in which it is possible to find arbitrarily large linearly independent subsets. These spaces do not have a finite basis.

EXAMPLE 6.29

Consider the space $\mathcal{C}$ of continuous functions on the real line. Define the polynomial functions by

$$\begin{aligned} p_0(x) &= 1, \\ p_1(x) &= x, \\ p_2(x) &= x^2, \\ &\vdots \\ p_n(x) &= x^n, \end{aligned}$$

for $x \in \mathbb{R}$.

Any collection of these functions is a linearly independent subset of $\mathcal{C}$, but none of them generates it. $\mathcal{C}$ therefore does not have a finite basis. ∎

6.5 Linear Spaces over the Binary Field

We have seen that the ring $\mathbb{Z}_2$, with addition and multiplication tables

$+$	0	1
0	0	1
1	1	0

$\times$	0	1
0	0	0
1	0	1

is the smallest example of a field. We will be using this field almost exclusively in the development of the theory of error-correcting codes. From now on, we will refer to it as the *binary field* and denote it by $\mathbb{B}$.

For any n, there are exactly 2^n n-tuples of elements of $\mathbb{B}^n$. For convenience, we will denote these simply by concatenating the bits, without commas in between or round brackets before and after them. (This also makes the elements of $\mathbb{B}^n$ look like sequences of bits.) Using these conventions, we have

$$\mathbb{B} = \{0, 1\},$$

$$\mathbb{B}^2 = \{00, 01, 10, 11\},$$

$$\mathbb{B}^3 = \{000, 001, 010, 011, 100, 101, 110, 111\},$$

$$\begin{aligned}\mathbb{B}^4 = \{&0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111,\\ &1000, 1001, 1010, 1011, 1100, 1101, 1110, 1111\},\end{aligned}$$

and so on.

We can define addition on $\mathbb{B}^n$ component-wise, so that, for example, in $\mathbb{B}^4$,

$$0011 + 0101 = 0110,$$

while multiplication by elements of $\mathbb{B}$ is defined very simply by setting $0b = 0$ and $1b = b$ for all $b \in \mathbb{B}^n$. These operations make $\mathbb{B}^n$ into a linear space over $\mathbb{B}$.

We can represent the members of $\mathbb{B}^n$ as the vertices of the unit cube in n-dimensional space.

The definitions of linear combinations, linear independence, bases and dimension given above for general linear spaces all apply to linear spaces over $\mathbb{B}$. A linear subspace of $\mathbb{B}^n$ of dimension k has exactly 2^k elements.

EXAMPLE 6.30

The following are linear subspaces of $\mathbb{B}^4$:

$$\{0000\}$$

$$\{0000, 1111\}$$

$$\{0000, 1000, 0010, 1010\}$$

$$\{0000, 1100, 0110, 0011, 1010, 1111, 0101, 1001\}.$$

Their dimensions are zero, one, two and three, respectively.

The following are not linear subspaces of $\mathbb{B}^4$:

$$\{0101\}$$

$$\{0101, 1010\}$$

$$\{0101, 1010, 1111\}$$

$$\{1000, 0100, 0010, 0001\}$$

$$\{0000, 1000, 0010, 1010, 1111\}.$$

□

DEFINITION 6.21 Coset *If L is a linear subspace of $\mathbb{B}$, and $x \in \mathbb{B}$, then the set of vectors*

$$x + L = \{x + y : y \in L\} \tag{6.9}$$

is a coset *of L.*

Cosets are also known as *affine subspaces*. $x + L = L$ if and only if $x \in L$, and $y + L = z + L$ if and only if $y \in z + L$. In particular, $0 + L = L$ as 0 always belongs to L.

EXAMPLE 6.31

$L = \{00, 11\}$ is a subspace of $\mathbb{B}^2$. The cosets of L are:

$$00 + L = 11 + L = L,$$

$$01 + L = 10 + L = \{01, 10\}.$$

□

EXAMPLE 6.32

$L = \{000, 111\}$ is a subspace of $\mathbb{B}^3$. The cosets of L are:

$$000 + L = 111 + L = L,$$

$$001 + L = 110 + L = \{001, 110\},$$

$$010 + L = 101 + L = \{010, 101\},$$

$$100 + L = 011 + L = \{100, 011\}.$$

□

EXAMPLE 6.33

$L = \{000, 101, 010, 111\}$ is a subspace of $\mathbb{B}^3$. The cosets of L are:

$$000 + L = 101 + L = 010 + L = 111 + L = L,$$

$$001 + L = 100 + L = 011 + L = 110 + L = \{001, 100, 011, 110\}.$$

□

DEFINITION 6.22 Weight *The* weight *of an element of $\mathbb{B}^n$ is the number of ones in it.*

We repeat the following definition from Chapter 5.

DEFINITION 6.23 Hamming Distance *The* Hamming distance *between x and $y \in \mathbb{B}^n$ is the number of places where they differ.*

The Hamming distance between x and y is equal to the weight of $x - y$.

EXAMPLE 6.34

The following table shows the weights of the elements of $\mathbb{B}^2$.

Element	*Weight*
00	0
01	1
10	1
11	2

The following table shows the Hamming distance between the pairs of elements of $\mathbb{B}^2$.

	00	01	10	11
00	0	1	1	2
01	1	0	2	1
10	1	2	0	1
11	2	1	1	0

□

EXAMPLE 6.35

The following table shows the weights of the elements of $\mathbb{B}^3$.

Element	*Weight*
000	0
001	1
010	1
011	2
100	1
101	2
110	2
111	3

The following table shows the Hamming distance between the pairs of elements of $\mathbb{B}^3$.

	000	001	010	011	100	101	110	111
000	0	1	1	2	1	2	2	3
001	1	0	2	1	2	1	3	2
010	1	2	0	1	2	3	1	2
011	2	1	1	0	3	2	2	1
100	1	2	2	3	0	1	1	2
101	2	1	3	2	1	0	2	1
110	2	3	1	2	1	2	0	1
111	3	2	2	1	2	1	1	0

∎

6.6 Linear Codes

We construct error-correcting codes by finding subsets of $\mathbb{B}^n$ with desirable properties.

DEFINITION 6.24 Binary Block Code *A* binary block code *is a subset of* $\mathbb{B}^n$ *for some* n. *Elements of the code are called* code words.

EXAMPLE 6.36

We can encode the alphabet using a subset of $\mathbb{B}^5$. We let 0001 stand for A, let 0010 stand for B, and so on, until 11010 stands for Z. The remaining six elements of $\mathbb{B}^5$ are not included in the code. ∎

Such simple codes do not have error-correcting properties. We need to have codes with more structure.

DEFINITION 6.25 Linear Code *A* linear code *is a linear subspace of* $\mathbb{B}^n$.

Linear codes are subsets of $\mathbb{B}^n$ with a linear structure added. Because multiplication is trivial in linear spaces over a binary field, a binary code is a linear code if and only if the sum of two code words is also a code word.

DEFINITION 6.26 Minimum Distance *The* minimum distance *of a linear code is the minimum of the weights of the non-zero code words.*

The minimum distance of a linear code is the minimum of the Hamming distances between pairs of code words.

The relationships between the minimum distance of a code and its capabilities in respect of detecting and correcting and detecting errors that were discussed in Chapter 5 also hold for linear codes. The additional structure gives us systematic ways of constructing codes with good error detecting and error correcting properties. Details of such results can be found in Chapter 4 of [1], Chapter 7 of [4] and Chapter 4 of [5].

EXAMPLE 6.37

In $\mathbb{B}^2$, $\{00, 01\}$ and $\{00, 10\}$ are linear codes with minimum distance 1 while $\{00, 11\}$ is a linear code with minimum distance 2. □

EXAMPLE 6.38

The following two-dimensional codes in $\mathbb{B}^3$ have minimum distance 1:

$$\{000, 001, 010, 011\},$$
$$\{000, 001, 100, 101\},$$
$$\{000, 010, 100, 110\},$$
$$\{000, 001, 110, 111\},$$
$$\{000, 010, 101, 111\},$$
$$\{000, 100, 011, 111\}.$$

The code $\{000, 011, 101, 110\}$ has minimum distance 2. □

Let L be a linear code. If L is a k-dimensional subspace of $\mathbb{B}^n$, then we can find a basis for L consisting of k code words $b_1, b_2, \ldots, b_k$ in $\mathbb{B}^n$. Every code word in L is a linear combination of these basis code words. There is a convenient matrix notation for linear codes that uses this fact.

DEFINITION 6.27 Generator Matrix *A* generator matrix *for a linear code is a binary matrix whose rows are the code words belonging to some basis for the code.*

A generator matrix for a k-dimensional linear code in $\mathbb{B}^n$ is a $k \times n$ matrix whose rank is k. Conversely, any $k \times n$ binary matrix with rank k is a generator matrix for some code.

We can compute the code words from the generator matrix by multiplying it on the left by all the row vectors of dimension k.

EXAMPLE 6.39

The code $\{0000, 0001, 1000, 1001\}$ is a two-dimensional linear code in $\mathbb{B}^4$. $\{0001, 1000\}$ is a basis for this code, which gives us the generator matrix

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

To find the code words from the generator matrix, we perform the following matrix multiplications:

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

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

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

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

This gives us the four code words, $0000, 1000, 0001$ and 1001, with which we started.

□

EXAMPLE 6.40

$\{0000, 0011, 0110, 1100, 0101, 1111, 1010, 1001\}$ is a three-dimensional linear code in $\mathbb{B}^4$.

$\{0011, 0110, 1100\}$ is a basis for this code, which gives us the generator matrix

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

To recover the code words from the generator matrix, we perform the following matrix multiplications:

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

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

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

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

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

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

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

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

$\{0101, 1001, 1010\}$ is also a basis for the code. It gives us the generator matrix

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

The code words can also be recovered from this matrix, but in a different order.

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

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

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

$$[0\ 1\ 1]\,G' = [0\ 0\ 1\ 1],$$

$$[1\ 0\ 0]\,G' = [0\ 1\ 0\ 1],$$

$$[1\ 0\ 1]\,G' = [1\ 1\ 1\ 1],$$

$$[1\ 1\ 0]\,G' = [1\ 1\ 0\ 0],$$

$$[1\ 1\ 1]\,G' = [0\ 1\ 1\ 0].$$

□

EXAMPLE 6.41

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

is a 3×5 binary matrix of rank 3. The code words of the three-dimensional linear code for which G is a generator matrix can be found by the following matrix multiplications.

$$[0\ 0\ 0]\,G = [0\ 0\ 0\ 0\ 0],$$

$$[0\ 0\ 1]\,G = [1\ 1\ 1\ 1\ 1],$$

$$[0\ 1\ 0]\,G = [0\ 0\ 1\ 1\ 1],$$

$$[0\ 1\ 1]\,G = [1\ 1\ 0\ 0\ 0],$$

$$[1\ 0\ 0]\,G = [0\ 0\ 0\ 0\ 1],$$

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

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

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

The code is

$$\{00000, 00001, 00110, 00111, 11000, 11001, 11110, 11111\}.$$

□

Because a linear space can have many bases, a linear code can have many generator matrices. This raises the question of when two generator matrices generate the same code.

DEFINITION 6.28 Elementary Row Operation *An* elementary row operation *on a binary matrix consists of replacing a row of the matrix with the sum of that row and any other row.*

If we have a generator matrix G for a linear code L, all other generator matrices for L can be obtained by applying a sequence of elementary row operations to G.

EXAMPLE 6.42

In a previous example, we saw that the linear code

$$\{0000, 0011, 0110, 1100, 0101, 1111, 1010, 1001\}$$

has the following generator matrices:

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

and

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

We can change G to G' by the following sequence of elementary row operations.

1. Replace the third row with the sum of the second and third rows. This gives

$$G_1 = \begin{bmatrix} 0\,0\,1\,1 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{bmatrix}.$$

2. Replace the first row with the sum of the first and second rows. This gives

$$G_2 = \begin{bmatrix} 0\,1\,0\,1 \\ 0\,1\,1\,0 \\ 1\,0\,1\,0 \end{bmatrix}.$$

3. Replace the second row with the sum of the first and second rows. This gives

$$G_3 = \begin{bmatrix} 0\,1\,0\,1 \\ 0\,0\,1\,1 \\ 1\,0\,1\,0 \end{bmatrix}.$$

4. Finally, replace the second row with the sum of the second and third rows. This gives

$$G' = \begin{bmatrix} 0\,1\,0\,1 \\ 1\,0\,0\,1 \\ 1\,0\,1\,0 \end{bmatrix}.$$

□

For error-correcting purposes, two codes that have the same minimum distance have the same properties. One way in which we can change a linear code without changing its minimum distance is to change the order of the bits in all the code words.

DEFINITION 6.29 Equivalent Codes *Two codes are* equivalent *if each can be constructed from the other by reordering the bits of each code word in the same way.*

The generator matrices of equivalent codes can be obtained from each other by interchanging columns.

DEFINITION 6.30 Canonical Form *The generator matrix G of a k-dimensional linear code in $\mathbb{B}^n$ is in* canonical form *if it is of the form*

$$G = [I : A],$$

where I is a $k \times k$ identity matrix and A is an arbitrary $k \times (n-k)$ binary matrix.

It is possible to convert the generator matrix for a code using elementary row operations (which do not change the set of code words) and column interchanges into the generator matrix of an equivalent code in the canonical form. The code words derived from a generator matrix in canonical form are also said to be in canonical form or in *systematic form*.

If the generator matrix G is in canonical form, and $\mathbf{w}$ is any k-bit word, the code word $\mathbf{s} = \mathbf{w}G$ is in systematic form and the first k bits of $\mathbf{s}$ are the same as the bits of $\mathbf{w}$.

EXAMPLE 6.43

We have seen in a previous example that

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

is a generator matrix for the code $\{0000, 0001, 1000, 1001\}$.

To reduce this to canonical form, we first interchange the first and second columns to get

$$G_1 = \begin{bmatrix} 0\ 0\ 0\ 1 \\ 0\ 1\ 0\ 0 \end{bmatrix}$$

and then interchange the first and last columns to get

$$G_2 = \begin{bmatrix} 1\ 0\ 0\ 0 \\ 0\ 1\ 0\ 0 \end{bmatrix}.$$

This is now in canonical form, with

$$A = \begin{bmatrix} 0\ 0 \\ 0\ 0 \end{bmatrix}.$$

The code generated by the canonical form of the generator matrix is

$$\{0000, 1000, 0100, 1100\}.$$

Note that both codes have minimum distance 1.

□

EXAMPLE 6.44

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

is a generator matrix of the linear code $\{0000, 0011, 0110, 1100, 0101, 1111, 1010, 1001\}$.

To reduce G to canonical form, we begin by applying the elementary row operation of replacing the second row of G by the sum of the first and second rows to give

$$G_1 = \begin{bmatrix} 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1 \\ 1\ 1\ 0\ 0 \end{bmatrix}.$$

Next we replace the third row by the sum of the second and third rows to give

$$G_2 = \begin{bmatrix} 0\ 0\ 1\ 1 \\ 0\ 1\ 0\ 1 \\ 1\ 0\ 0\ 1 \end{bmatrix}.$$

Finally we interchange the first and third columns to give

$$G_3 = \begin{bmatrix} 1\ 0\ 0\ 1 \\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 1 \end{bmatrix}.$$

This is in canonical form with

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

The code generated by the canonical form of G is the same as the code generated by G. □

EXAMPLE 6.45

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

is a generator matrix of the linear code

$$\{00000, 00001, 00110, 00111, 11000, 11001, 11110, 11111\}.$$

To reduce G to canonical form we start by replacing the third row with the sum of the second and third rows to give

$$G_1 = \begin{bmatrix} 0\ 0\ 0\ 0\ 1 \\ 0\ 0\ 1\ 1\ 1 \\ 1\ 1\ 0\ 0\ 0 \end{bmatrix}.$$

Next, we replace the second row with the sum of the first and second rows to give

$$G_2 = \begin{bmatrix} 0\,0\,0\,0\,1 \\ 0\,0\,1\,1\,0 \\ 1\,1\,0\,0\,0 \end{bmatrix}.$$

We interchange the first and last columns, obtaining

$$G_3 = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1 \end{bmatrix},$$

and finally interchange the second and third columns, to get

$$G_4 = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1 \end{bmatrix}.$$

G_4 is now in canonical form, with

$$A = \begin{bmatrix} 0\,0 \\ 1\,0 \\ 0\,1 \end{bmatrix}.$$

It generates the code

$$\{00000, 10000, 01010, 00101, 11010, 10101, 01111, 11111\}.$$

□

DEFINITION 6.31 Parity Check Matrix *The* parity check matrix *of a linear code with $k \times n$ generator matrix G is the $k \times n$ matrix H satisfying*

$$GH^T = 0, \tag{6.10}$$

where H^T is the transpose of H and 0 denotes the $k \times (n-k)$ zero matrix.

If G is in canonical form, $G = [I : A]$, with I the $k \times k$ identity matrix, then $H = [A^T : I]$, where I is the $(n-k) \times (n-k)$ identity matrix. (This is true if G and H are binary matrices. In general, we should have $H = [-A^T : I]$. In the binary case, $A = -A$.) If G is not in canonical form, we can find H by reducing G to canonical form, finding the canonical form of H using the equation above, and then reversing the column operations used to convert G to canonical form to convert the canonical form of H to the parity check matrix of G.

EXAMPLE 6.46

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

is in canonical form with

$$A = \begin{bmatrix} 0\,1 \\ 1\,1 \\ 1\,0 \end{bmatrix}.$$

H is obtained by transposing A and adjoining a 2×2 identity matrix to get

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

As expected, we have

$$GH^T = \begin{bmatrix} 1\,0\,0\,0\,1 \\ 0\,1\,0\,1\,1 \\ 0\,0\,1\,1\,0 \end{bmatrix} \begin{bmatrix} 0\,1 \\ 1\,1 \\ 1\,0 \\ 1\,0 \\ 0\,1 \end{bmatrix} = \begin{bmatrix} 0\,0 \\ 0\,0 \\ 0\,0 \end{bmatrix}.$$

□

EXAMPLE 6.47

We have seen in a previous example that

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

is a generator matrix of the linear code

$$\{00000, 00001, 00110, 00111, 11000, 11001, 11110, 11111\},$$

which can be reduced to the canonical form

$$G_c = \begin{bmatrix} 1\,0\,0\,0\,0 \\ 0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1 \end{bmatrix}$$

by the following operations:

1. Replace the third row by the sum of the second and third rows.

2. Replace the second row by the sum of the first and second rows.

3. Interchange the first and fifth columns.

4. Interchange the second and third columns.

The canonical form of the parity check matrix is

$$H_c = \begin{bmatrix} 0\,1\,0\,1\,0 \\ 0\,0\,1\,0\,1 \end{bmatrix}.$$

We have

$$G_c H_c^T = \begin{bmatrix} 0\,0 \\ 0\,0 \\ 0\,0 \end{bmatrix}.$$

To find the parity check matrix of G, we apply the column operations used to reduce G to canonical form to H_c in reverse order. We start by interchanging the second and third columns to get

$$H_1 = \begin{bmatrix} 0\,0\,1\,1\,0 \\ 0\,1\,0\,0\,1 \end{bmatrix}.$$

We interchange the first and fifth columns to get

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

We now check that

$$GH^T = \begin{bmatrix} 0\,0 \\ 0\,0 \\ 0\,0 \end{bmatrix}.$$

□

EXAMPLE 6.48

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

is a generator matrix of the linear code $\{000, 101, 011, 110\}$. It is in canonical form; so the parity check matrix is

$$H = \begin{bmatrix} 1\,1\,1 \end{bmatrix}.$$

Let us see what happens to elements of $\mathbb{B}^3$ when they are multiplied by H^T:

$$\begin{aligned}
000H^T &= 0\\
001H^T &= 1\\
010H^T &= 1\\
011H^T &= 0\\
100H^T &= 1\\
101H^T &= 0\\
110H^T &= 0\\
111H^T &= 1.
\end{aligned}$$

The product of H and any of the code words in the code generated by G is zero, while the remaining products are non-zero. □

The example above should not be surprising, for we have the following result.

RESULT 6.1

If L is a k-dimensional linear code in $\mathbb{B}^n$, and G is a generator matrix for L, every code word in L can be obtained by taking some $b \in \mathbb{B}^k$ and multiplying it by G, to get the code word bG. If we now multiply this by the transpose of the parity check matrix H, we get

$$(bG)H^T = b(GH^T) = b0 = 0. \tag{6.11}$$

The parity check matrix gives us an easy way of determining if a word $w \in \mathbb{B}^n$ belongs to the linear code L: we compute wH^t. If the result is the zero matrix, w is a code word. If not, then w is not a code word. As we shall see, the parity check matrix enables us to do more than just check if a word is a code word.

EXAMPLE 6.49

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

is a generator matrix of the linear code $\{0000, 0011, 1100, 1111\}$. We reduce it to canonical form by interchanging the first and last columns to get

$$G_c = \begin{bmatrix} 1\,0\,1\,0 \\ 0\,1\,0\,1 \end{bmatrix},$$

which generates the code $\{0000, 1010, 0101, 1111\}$.

The canonical form of the parity check matrix is

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

If we look at the products

$$\begin{aligned}
0000H_c^T &= 00 \\
0001H_c^T &= 01 \\
0010H_c^T &= 10 \\
0011H_c^T &= 11 \\
0100H_c^T &= 01 \\
0101H_c^T &= 00 \\
0110H_c^T &= 11 \\
0111H_c^T &= 10 \\
1000H_c^T &= 10 \\
1001H_c^T &= 11 \\
1010H_c^T &= 00 \\
1011H_c^T &= 01 \\
1100H_c^T &= 11 \\
1101H_c^T &= 10 \\
1110H_c^T &= 01 \\
1111H_c^T &= 00
\end{aligned}$$

we see that only the code words in the code generated by G_c have products equal to 00.

If we now interchange the first and last columns of H_c to get the parity check matrix of G, we get

$$H = \begin{bmatrix} 0\ 0\ 1\ 1 \\ 1\ 1\ 0\ 0 \end{bmatrix},$$

and the products are

$$\begin{aligned}
0000H^T &= 00 \\
0001H^T &= 10 \\
0010H^T &= 10 \\
0011H^T &= 00 \\
0100H^T &= 01 \\
0101H^T &= 11 \\
0110H^T &= 11
\end{aligned}$$

$$0111H^T = 01$$
$$1000H^T = 01$$
$$1001H^T = 11$$
$$1010H^T = 11$$
$$1011H^T = 01$$
$$1100H^T = 00$$
$$1101H^T = 10$$
$$1110H^T = 10$$
$$1111H^T = 00$$

and again only the code words in the code generated by G have products equal to 00.

In this example, we have $G = H$ and $G_c = H_c$. A generator matrix can be the same as its parity check matrix. □

(Although we have not introduced any terminology regarding linear transformations, readers who are familiar with this topic will realise that a linear code L is the range of the linear transformation determined by the generator matrix, and the kernel of the linear transformation determined by the transpose of the associated parity check matrix.)

6.7 Encoding and Decoding

The use of a linear code for error correction involves an encoding step to add redundancy and a later decoding step which attempts to correct errors before it removes the redundancy. This is essentially the same process as was described in Chapter 5, but the additional structure of linear codes enables us to find new algorithms for the encoding and decoding processes.

In the simplest case, the encoding step uses a generator matrix G of a linear code, while the decoding step uses the corresponding parity check matrix.

The encoding process takes a string, $\mathbf{b}$, k bits in length and produces a code word, $\mathbf{w}$, n bits in length, $\mathbf{w} = \mathbf{b}G$. The first k bits of $\mathbf{w}$ are the *message bits*, and the remaining bits are the *check bits*. The latter represent the redundancy which has been added to the message.

If G is in canonical form, the first k bits of $\mathbf{w}$ are the bits of $\mathbf{b}$ and the code is said to be in *systematic form*. In this case, code words can be decoded by simply removing the last $(n - k)$ bits.

Suppose that a code word is corrupted by noise between encoding and decoding. One or more bits will be changed, zeros becoming ones and ones becoming zeros. It is possible that the result will be another code word, in which case it will not be possible to detect that corruption has occurred. Otherwise, the corrupted string will not be a code word, and attempts may be made to restore the uncorrupted code word as part of the decoding process.

The decoding process therefore has two stages. If the received string is not a code word, the uncorrupted code word has to be restored. The code word is then decoded to recover the original string.

If we assume that the corruption occurred while transmission through a *binary symmetric channel*, and use the *maximum likelihood decoding strategy*, it follows that a corrupted string should be restored to the code word which is closest to it (in terms of the Hamming distance). (Other assumptions about the characteristics of the noise process may lead to other procedures.)

We suppose that the generator matrix G generates the linear code L, and that the code word $\mathbf{w}$ is corrupted by a noise vector $\mathbf{e}$, and the result is the vector $\mathbf{x} = \mathbf{w} + \mathbf{e}$. If $\mathbf{x}$ is not a code word, the decoder must find the closest code word $\mathbf{y}$, or equivalently, the error vector of smallest weight.

If $\mathbf{x}$ is not a code word, it belongs to the coset $\mathbf{x} + L$; so we assume that $\mathbf{x}$ is of the form $\mathbf{e} + \mathbf{u}$ where $\mathbf{u} \in L$ and $\mathbf{e}$ is the vector of least weight in the coset.

DEFINITION 6.32 Syndrome *If L is a linear code with generator matrix G and parity check matrix H, the* syndrome *of* $\mathbf{x}$ *is given by* $s = \mathbf{x}H^T$.

Members of the same coset of L have the same syndrome, so there is a one-to-one correspondence between cosets and syndromes.

We now have a procedure for removing noise. Draw up the *syndrome table*, which is a list of the cosets, showing the vector of minimum weight and the syndrome. When a string that is not a code word is received, compute its syndrome. Add the vector of minimum weight from the corresponding coset to the corrupted string to recover the uncorrupted code word.

EXAMPLE 6.50

Suppose that

$$H = \begin{bmatrix} 0\,0\,1\,1 \\ 1\,1\,0\,0 \end{bmatrix}$$

is the parity check matrix of the linear code L.

To draw up the syndrome table, we start with the elements of L itself, which have syndrome 00. The first row of the syndrome table looks like

$$00|0000\ 0011\ 1100\ 1111|0000$$

where the syndrome appears at the left, the code words in L appear in the middle and the vector of minimum weight appears on the right.

We now choose a vector that does not belong to L and compute its syndrome. Let us choose 0001, whose syndrome is 10. We add this vector to all the code words in L to get the next row of the syndrome table:

$$\begin{array}{l|l|l} 00 & 0000\ 0011\ 1100\ 1111 & 0000 \\ 10 & 0001\ 0010\ 1101\ 1110 & 0001 \end{array}$$

Note that we have two vectors of minimum weight in this coset. We have chosen one of them arbitrarily.

We now choose another vector and compute its syndrome. If we choose 0110, the syndrome is 11 and we can add another row to the syndrome table:

$$\begin{array}{l|l|l} 00 & 0000\ 0011\ 1100\ 1111 & 0000 \\ 10 & 0001\ 0010\ 1101\ 1110 & 0001 \\ 11 & 0110\ 0101\ 1010\ 1001 & 0101 \end{array}$$

Again, we have chosen the vector of minimum weight arbitrarily.

Again we choose a vector and compute its syndrome. The syndrome of 0100 is 01. We use this to complete the table:

$$\begin{array}{l|l|l} 00 & 0000\ 0011\ 1100\ 1111 & 0000 \\ 01 & 0100\ 0111\ 1000\ 1011 & 0100 \\ 10 & 0001\ 0010\ 1101\ 1110 & 0001 \\ 11 & 0110\ 0101\ 1010\ 1001 & 0101 \end{array}$$

We can now use the syndrome table for error correction. Suppose the vector 1010 is to be corrected. We compute its syndrome, which is 11. We add the vector of minimum weight in the coset with syndrome 11 to 1010,

$$1010 + 0101 = 1111.$$

So the corrected code word is 1111.

If the vector 0111 is to be corrected, its syndrome is 01, so we add

$$0111 + 0100 = 0011$$

to obtain the corrected code word 0011.

If the vector 1110 is to be corrected, its syndrome is 10, so we add

$$1110 + 0001 = 1111$$

to obtain the corrected code word 1111.

If the vector 1100 is to be corrected, its syndrome is 00, which means that it is a code word and no correction is necessary.

Note that the error correction process may not give the code word that was corrupted by noise. Suppose that the original code word is 0000 and that the leftmost bit gets corrupted, giving 1000. The correction procedure will produce 1100, not 0000. This kind of error is unavoidable if a coset has more than one vector of minimum weight.

□

6.8 Codes Derived from Hadamard Matrices

DEFINITION 6.33 Hadamard Matrix *A* Hadamard matrix *is an orthogonal matrix whose elements are either* 1 *or* -1.

The simplest Hadamard matrix is the 2×2 matrix

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

If H is an $n \times n$ Hadamard matrix, then the $2n \times 2n$ matrix

$$H' = \begin{bmatrix} H & H \\ H & -H \end{bmatrix}$$

is also a Hadamard matrix. This gives us a means of constructing $2^k \times 2^k$ Hadamard matrices for all $k \geq 1$.

EXAMPLE 6.51

We construct a 4×4 Hadamard matrix by combining the matrices H_2 above:

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

We construct a 8×8 Hadamard matrix by combining the matrices H_4:

$$H_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}.$$

We can continue in this way to construct 16×16 Hadamard matrices, 32×32 Hadamard matrices, and so on. ∎

$n \times n$ Hadamard matrices exist for n other than 2^m, but n must be a multiple of 4. Some results about the existence of Hadamard matrices for various values of n are cited in Section 5.7 of [5]. Finding Hadamard matrices is still a research topic.

If H is a $n \times n$ Hadamard matrix, we can construct a code with $2n$ code words, each n bits long, and distance $n/2$, in the following way. Let $v_1, v_2, \ldots, v_n$ denote the rows of H. For each v_i, we construct two code words. For the first, we change all the 1's to 0's and the -1's to 1; for the second, we do not change the 1's, but change the -1's to 0's.

If n is not a power of 2, the resulting code is not a linear code. If n is a power of 2 and the Hadamard matrix is constructed recursively as described above, the resulting code is a first-order Reed-Muller code. (Reed-Muller codes are discussed in the next chapter.)

EXAMPLE 6.52

We apply the construction described above to

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}.$$

The first row of H_2 is $[1\ 1]$, which gives us the code words 00 and 11.

The second row is $[1\ -1]$, which gives us the code words 01 and 10.

The code is $\{00, 01, 10, 11\}$, the whole of $\mathbb{B}^2$. ∎

EXAMPLE 6.53

We get a more interesting example if we use H_4.

The first row of H_4 is $[1\ 1\ 1\ 1]$, which gives us the code words 0000 and 1111.

The second row of H_4 is $[1\ -1\ 1\ -1]$, which gives us the code words 0101 and 1010.

The third row of H_4 is $[1\ 1\ -1\ -1]$, which gives us the code words 0011 and 1100.

The fourth row of H_4 is $[1\ -1\ -1\ 1]$, which gives us the code words 0110 and 1001.

The code is $\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$. □

The use of Hadamard matrices gives good separation between code words, which results in good error-correcting capabilities.

6.9 Exercises

1. Find all the homomorphisms from $\mathbb{Z}_2$ into the two groups of order 4.

*2. Find all the continuous homomorphisms from the group of real numbers with the operation of addition to the group of positive real numbers with the operation of multiplication.

3. Find the values of the following expressions when the numbers and operations belong to the specified Cyclic Ring:

   (a) $(1+2)\times(2+3)$ in $\mathbb{Z}_4$;
   (b) $(1+2)\times(2+3)$ in $\mathbb{Z}_5$;
   (c) $(1+2)\times(2+3)$ in $\mathbb{Z}_6$;
   (d) $(1+3)\times(2+4)$ in $\mathbb{Z}_5$;
   (e) $(1+3)\times(2+4)$ in $\mathbb{Z}_6$;
   (f) $(1+3)\times(2+4)$ in $\mathbb{Z}_7$.

4. Which of the following subsets of $\mathbb{R}^4$ are linearly independent?

   (a) $\{(1,-1,1,-1)\}$;
   (b) $\{(1,-1,1,-1),(-1,1,-1,1)\}$;
   (c) $\{(1,-1,1,-1),(1,1,1,1)\}$;
   (d) $\{(1,1,1,1),(-1,1,-1,1),(0,2,0,2)\}$;
   (e) $\{(1,1,1,1),(-1,1,-1,1),(0,2,0,3)\}$;
   (f) $\{(1,1,1,1),(-1,1,-1,1),(0,0,0,0)\}$.

5. Which of the following sets are bases for $\mathbb{R}^3$?

   (a) $\{(1,0,0),(0,0,1)\}$;
   (b) $\{(2,-3,4),(-4,5,-6)\}$;
   (c) $\{(1,1,0),(1,-1,0),(0,0,1)\}$;
   (d) $\{(1,1,1),(1,-1,0),(0,0,1)\}$;
   (e) $\{(1,0,0),(0,1,0),(0,0,1),(1,1,1)\}$.

6. Which of the following sets are linear subspaces of $\mathbb{B}^5$?

   (a) $\{00000\}$;
   (b) $\{11111\}$;
   (c) $\{00000, 11111\}$;
   (d) $\{01010, 10101\}$;
   (e) $\{00000, 01010, 10101\}$;
   (f) $\{00000, 01010, 10101, 11111\}$;
   (g) $\{10000, 01000, 00100, 00010, 00001\}$;
   (h) $\{00000, 10000, 01000, 00100, 00010, 00001\}$;
   (i) $\{00000, 00100, 01010, 10001, 01110, 10101, 11011, 11111\}$.

7. Determine which of the following subsets of $\mathbb{B}^4$ are linear subspaces and write down all the cosets of those which are.

   (a) $\{0000, 1100, 0011, 1001\}$;
   (b) $\{0000, 1100, 0011, 1111\}$;
   (c) $\{0000, 1000, 0010, 1010\}$;
   (d) $\{1100, 0011, 1001, 1111\}$.

8. Construct a linear subspace of $\mathbb{B}^6$ of dimension 3 and find the weight of each of its elements.

9. Construct a table showing the Hamming distance between each pair of elements in $\mathbb{B}^4$.

10. Find the dimension and the minimum distance of each of the following linear codes in $\mathbb{B}^4$ or $\mathbb{B}^5$.

    (a) $\{0000, 1000, 0001, 1001\}$;
    (b) $\{0000, 1010, 0101, 1111\}$;
    (c) $\{0000, 1000, 0110, 1110\}$;
    (d) $\{0000, 1100, 0110, 0011, 1001, 1010, 0101, 1111\}$;
    (e) $\{00000, 10101, 01010, 11111\}$;

(f) $\{00000, 01000, 00010, 01010\}$;

(g) $\{00000, 11000, 01110, 00011, 10110, 11011, 01101, 10101\}$.

11. Write down a generator matrix for each of the linear codes in the previous exercise.

12. Find the code words of the linear codes defined by the following generator matrices:

(a)
$$G = \begin{bmatrix} 0\,0\,1 \\ 1\,1\,0 \end{bmatrix};$$

(b)
$$G = \begin{bmatrix} 0\,1\,1\,0 \\ 1\,1\,0\,1 \end{bmatrix};$$

(c)
$$G = \begin{bmatrix} 1\,1\,0\,0 \\ 0\,1\,1\,0 \\ 0\,0\,1\,1 \end{bmatrix};$$

(d)
$$G = \begin{bmatrix} 1\,1\,1\,0\,0 \\ 0\,0\,0\,1\,1 \end{bmatrix};$$

(e)
$$G = \begin{bmatrix} 1\,1\,0\,0\,0 \\ 1\,0\,1\,0\,1 \\ 0\,0\,0\,1\,1 \end{bmatrix};$$

(f)
$$G = \begin{bmatrix} 1\,1\,1\,1\,0 \\ 1\,1\,1\,0\,1 \\ 1\,1\,0\,1\,1 \\ 1\,0\,1\,1\,1 \end{bmatrix}.$$

13. Reduce the generator matrices in the previous exercise to canonical form and find their parity check matrices.

14. Find the linear codes defined by the following parity check matrices:

(a)

$$H = \begin{bmatrix} 1\ 1\ 0 \\ 0\ 1\ 1 \end{bmatrix};$$

(b)

$$H = \begin{bmatrix} 1\ 0\ 0\ 1 \\ 0\ 1\ 0\ 1 \\ 0\ 0\ 1\ 1 \end{bmatrix};$$

(c)

$$H = \begin{bmatrix} 1\ 1\ 0\ 0 \\ 0\ 1\ 1\ 0 \\ 0\ 0\ 1\ 1 \end{bmatrix};$$

(d)

$$H = \begin{bmatrix} 1\ 1\ 1\ 0\ 0 \\ 0\ 0\ 1\ 1\ 1 \end{bmatrix};$$

(e)

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

15. For each of the parity check matrices in the previous exercise, construct the syndrome table and use it to decode the following:

    (a) $100, 010, 001, 111$;

    (b) $1000, 0100, 0010, 0001$;

    (c) $1110, 1101, 1011, 0111$;

    (d) $10101, 01010, 11111$;

    (e) $10101, 01010, 11111$.

*16. Suppose that the code words of a k-dimensional linear code of length n are arranged as the rows of a matrix with 2^k rows and n columns, with no column consisting only of 0s. Show that each column consists of 2^{k-1} 0s and 2^{k-1} 1s. Use this to show that the sum of the weights of the code words is $n2^{k-1}$.

*17. Use the result of the previous Exercise to show that the minimum distance of a k-dimensional linear code of length n is no more than $n2^{k-1}/(2^k - 1)$.

*18. Show that a k-dimensional linear code of length n whose minimum distance is more than $2m$ must have at least $\log_2(1 + C_1^n + C_2^n + \ldots + C_m^n)$ parity check bits, where C_p^n denotes the number of combinations of p objects taken from a collection of n objects.

*19. Suppose we have the codes

$$\{000, 011, 101, 110\}$$

in which the third bit is the parity of the first two, and

$$\{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\}$$

where the fourth bit is the parity of the first three. We can use these to construct a code of length 12 with 6 information bits and 6 parity check bits by arranging the 6 information bits in an array with 2 rows and 3 columns, adding the parity check bits of the rows and the columns and completing the array with the parity of the parity row (or column). For example, the word 000111 is arranged as

$$\begin{array}{c} 0\ 0\ 0 \\ 1\ 1\ 1 \end{array}$$

the parity check bits are added to the rows and columns to give

$$\begin{array}{c} 0\ 0\ 0\ 0 \\ 1\ 1\ 1\ 1 \\ 1\ 1\ 1 \end{array}$$

and adding the final parity check bit

$$\begin{array}{c} 0\ 0\ 0\ 0 \\ 1\ 1\ 1\ 1 \\ 1\ 1\ 1\ 1 \end{array}$$

completes the array. The rows are then concatenated to give the code word 000011111111. Write down the generator matrix of this code.

*20. The construction of the previous Exercise, which is due to Elias [2], can be applied to any pair of codes of length m and n to produce a code of length mn. Show that the minimum distance of the resulting code is equal to the product of the minimum distances of the two codes used in the construction.

6.10 References

[1] R. Ash, *Information Theory,* John Wiley & Sons, New York, 1965.

[2] P. Elias, Error-free coding, *Transactions of the I.R.E. Professional Group on Information Theory*, PGIT-4, 29–37, September 1954.

[3] J. B. Fraleigh, *A First Course in Abstract Algebra,* 5th ed., Addison-Wesley, Reading, MA, 1994.

[4] R. J. McEliece, *The Theory of Information and Coding,* 2nd ed., Cambridge University Press, Cambridge, 2002.

[5] W. W. Peterson, *Error-correcting Codes,* MIT Press, Cambridge, MA, 1961.