# MATH3302

Coding and Cryptography

Coding Theory

2010

# Contents

# 1 Introduction to coding theory

## 1.1 Introduction

*Coding theory* is the study of methods for efficient and accurate transfer of information from one place to another. It is different to cryptography: we are no longer interested in secrecy, just accuracy and efficiency.

Coding theory has many uses: minimising noise on CD players, data transfer on phone lines or the internet, ethernet connections, data transfer from memory to CPU in a computer and space communication.

When transfering information from one place to another the information passes through a *channel*. The channel is the physical medium through which the information is transferred, for example the atmosphere or a phone line. Errors in the transferred information occur due to *noise* on the channel, that is, undesirable disturbances which may cause information received to differ from information transmitted. Noise can be caused by many things, for example, sunspots, lightning, poor typing, poor hearing.

Coding theory deals with the problem of *detecting* and *correcting* transmission errors caused by noise on the channel. The primary goals are to provide:

1. fast encoding of information;
2. easy transmission of encoded messages;
3. fast decoding of received message;
4. detection and correction of errors introduced in the channel;
5. maximum transfer of information per unit time.

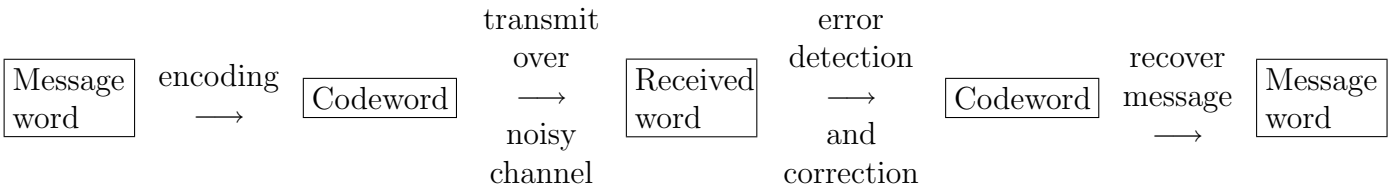
These goals are not necessarily compatible! Goal 4 is where we will spend most of our discussion.

**Example 1.1** Consider normal conversation (codewords are english words, channel is atmosphere, encoder is speech, decoder is hearing). We have in-built error correction: if you received the message "apt natural, i have a gub", you would probably know what is meant. You can use the redundancy inherent in the message to infer its meaning.

*Redundancy* is a fundamental component of coding theory. We will be adding extra bits of information to each word before transmission in order to (hopefully) allow the effect of noise to be countered, and the correct word to be inferred. The challenge is to add as little extra information as possible, while still achieving the desired level of error detection and correction.

**Example 1.2** *A 3-fold repetition code:* Suppose we have four message words: 00, 01, 10, 11, and we encode them by repeating them three times to get the four codewords 000000, 010101, 101010, 111111. After transmission, when we receive a word of length 6 we apply the decoding process of choosing the codeword which is closest to the received word. This system allows the detection of an error in up to two positions and the correction of an error in one position.

**Summary of information transmission process:**




**Example 1.3** *The ISBN code:* Every recent book should have an International Standard Book Number (ISBN). This is a 10-digit codeword $x_1x_2x_3x_4x_5x_6x_7x_8x_9x_{10}$ assigned by the publisher. The first digit indicates the language (0 for English) and the second and third digits indicate the publisher (for example, 19 stands for Oxford University Press). The next 6 digits are the book number assigned by the publisher. The final digit is a check digit, and is chosen so that the sum

$$\sum_{i=1}^{10} ix_i$$

is divisible by 11. (Note that the symbol X is used for the final digit to represent the number 10.) This system allows the detection of two types of errors: a single incorrect digit, and the transposition of two digits.

## 1.2 Basic definitions and assumptions

**Definition 1.4** A *q-ary code* is a set of sequences of symbols where each symbol is chosen from a set of $q$ distinct elements. The set of $q$ distinct elements is called the *alphabet*. A sequence of symbols is called a *word* and a word that occurs in the code is called a *codeword*. The *length* of a word is the number of symbols in the word. A code in which each codeword has the same length, $n$ say, is called a *block code of length* $n$. The number of codewords in a code $C$ is denoted by $|C|$.

**Example 1.5** The ISBN code is an 11-ary block code of length 10 based on the alphabet $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X\}$.

**Definition 1.6** 2-ary codes are called *binary* codes, and are usually based on the alphabet $\{0, 1\}$. In this course, we focus on binary block codes, so the word code will refer to a binary block code unless otherwise indicated.

**Definition 1.7** A binary codeword is transmitted by sending its digits, one at a time and in order, across a *binary channel*. A binary channel is *symmetric* if 0 and 1 are transmitted with equal accuracy; that is, the probability of receiving the correct digit is independent of whether the digit transmitted was 0 or 1. The *reliability* of a binary symmetric channel (BSC) is a real number $0 \leq p \leq 1$, where $p$ is the probability that the digit sent is the digit received. The channel is *perfect* if $p = 1$.

We need to make a number of assumptions about the binary channel.

- We assume that any codeword of length $n$ containing 0s and 1s will be received as a word of length $n$ containing 0s and 1s, although not necessarily the same as the original codeword. We also assume that it is easy to identify the beginning of the first word received, and hence that we can identify each received word.
- We assume that noise on a channel is scattered randomly, rather than in bursts. Thus, the probability of any one digit being altered by noise is equal to the probability of any other digit being altered. (This is perhaps not reasonable, but we will relax this later on.)
- We assume that no channel is perfect.
- If $p$ is the probability that the digit received is the same as that sent, then $1-p$ is the probability that the digit received **is not** the same as the digit sent. If any channel has $p < 0.5$, this can be converted to a channel with $0.5 \leq p \leq 1$ (by inverting each digit), so we will assume that we are communicating via a symmetric binary channel with $0.5 \leq p < 1$.

**Definition 1.8** Let $v$ be the codeword of length $n$ transmitted, $w$ be the word received and assume communication is over a BSC with reliability $p$ and with randomly scattered noise. Let $\Phi_p(v, w)$ be the probability that if $v$ is transmitted then $w$ is received. If $v$ and $w$ disagree in $d$ positions then we have

$$\Phi_p(v, w) = p^{n-d}(1-p)^d.$$

**Exercise 1.9** Let $C$ be a code of length 6, and suppose that we are transmitting codewords over a BSC with reliability $p = 0.95$ and with randomly scattered noise.

1. For any codeword $v \in C$, what is the probability that $v$ is received correctly?
2. Let $v = 101010 \in C$ and $x = 001010$. What is $\Phi_p(v, x)$?
3. Let $v = 101010 \in C$ and $w = 011010$. What is $\Phi_p(v, w)$?
4. For any codeword $v \in C$, what is the probability that a word is received which differs from $v$ in one position (so one error has occurred)?
5. For any codeword $v \in C$, what is the probability that a word is received which differs from $v$ in two positions (so two errors have occurred)?
6. For any codeword $v \in C$, what is the probability that a word is received which differs from $v$ in two or more positions?

If the reliability of the channel was only $p = 0.51$, then the answers to questions 1 – 3 above would be 0.0176, 0.0169 and 0.0162, respectively. It is vital to have a reliable channel in order to have any chance of receiving transmitted information correctly. In practice, channel reliability is usually higher than 0.95.

## 1.3 Introduction to error detection

Suppose a word is received, and this word is not a codeword. Then we have *detected* that at least one error has occurred in transmission.

**Example 1.10** Let $C_1 = \{00, 01, 10, 11\}$. Then $C_1$ cannot detect any errors in transmission.

**Example 1.11** Let $C_2$ be a new code formed by repeating each codeword of $C_1$, so

$$C_2 = \{0000, 0101, 1010, 1111\}.$$

If a single error occurs in any transmitted codeword $w$, then the received word will not be a codeword. This allows detection of any single error. If two errors occur in any transmitted codeword $w$, then the received word may or may not be a codeword. So some sets of two errors will be detected, but not all sets of two errors.

We call $C_2$ a *repetition* code: in fact, it is a 2-*fold repetition code*, denoted $Rep(2)$. An $n$-fold repetition code is formed by taking the $2^k$ words of length $k$ and forming the codewords of length $kn$ by writing down, $n$ times, each word of length $k$.

**Example 1.12** Let $C_3$ be a new code formed from $C_1$ by adding a third digit to each codeword so that the number of 1s in each codeword is even, so $C_3 = \{000, 011, 101, 110\}$. The added digit is called a *parity-check* digit, and it enables detection of any single error. This code will not detect any set of two errors in a transmitted codeword.

**Exercise 1.13** Consider a communication channel with reliability $p = 1 - 10^{-8}$. Consider the code $C$ consisting of all $2^{11}$ words of length 11 (so there are no check digits). Suppose that digits are transmitted at $10^7$ digits per second. On average, approximately how many words are (undetectedly) transmitted incorrectly per minute?

Now let $D$ be the code obtained from the code $C$ in Example 1.13 by adding a parity-check digit to each codeword of $C$, so that the number of 1s in each transmitted word is even. Using the same reliability and rate of transmission, we will determine how many words are (undetectedly) transmitted incorrectly per minute if the code $D$ is used. The code $D$ will detect a single error in a transmitted word, so the probability of an incorrect word being received and not detected is

$$1 - P(0 \text{ errors}) - P(1 \text{ error}) = 1 - \binom{12}{0} p^{12}(1-p)^0 - \binom{12}{1} p^{11}(1-p)^1 \approx 6.6 \times 10^{-15}.$$

Words are transmitted at a rate of

$$\frac{10^7 \text{ digits}}{1 \text{ second}} \times \frac{1 \text{ word}}{12 \text{ digits}} \times \frac{60 \text{ seconds}}{1 \text{ minute}} = 5 \times 10^7 \text{ words per minute.}$$

Thus approximately $3.3 \times 10^{-7}$ incorrect words are undetectedly transmitted per minute. That is approximately 1 word in 6 years.

By adding only a small amount of extra information (redundancy) we drastically reduced the number of incorrect words that slip through without detection.

## 1.4 Information rate of a code

In the previous subsection, we created two new codes, $C_2$ and $C_3$, by adding extra bits of information (redundancy) to the codewords of $C_1$, thus enabling error detection. In each case we added different amounts of information, but each code can still only detect a *single* error. In some sense, the code $C_3$ may be more efficient than the code $C_2$.

Clearly, by adding extra bits of information to the words, we can improve error detection. Of course, as check digits are added, more bits must be transmitted for each codeword, thereby increasing transmission time.

**Definition 1.14** Many codes are obtained by taking the $2^k$ words of length $k$ and adding $n-k$ *check bits* to each word, thus giving codewords of length $n$. A code of length $n$, with $2^k$ codewords is called an $(n, k)$ code. The number $k$ is the *dimension* of the code, and we say that such a code has $k$ *message bits*.

**Definition 1.15** If $C$ is any code of length $n$ then the *information rate* or *rate* of $C$ is given by

$$\frac{1}{n} \log_2 |C|.$$

Hence if $C$ is an $(n, k)$ code (so $|C| = 2^k$) then the information rate of $C$ is $k/n$.

**Exercise 1.16** Compare the information rates of the codes $C$ and $D$ from Exercise 1.13 and the discussion following it.

Thus, for a (reasonably) small reduction in efficiency/information rate, it is possible to incorporate extra information, allowing detection of a single error.

## 1.5 Introduction to error correction

What can be done if the existence of an error is detected? Requesting the retransmission of a message has a significant cost: we need to interrupt and delay transmission. It would be much better if we could not only detect the *existence* of an error, but could also *locate* it. If we can locate where the error occurred, then we can *correct* it by inverting the received bit.

**Example 1.17** Let $Rep(3)$ be the 3-fold repetition code formed by writing down each word of length 2 three times, so $Rep(3) = \{000000, 010101, 101010, 111111\}$. If a single error occurs in any digit of any transmitted codeword $v$, then the received word $w$ will not be a codeword. This allows detection of a single error. Moreover, the received word must have originated from one of the codewords. It differs from one codeword ($v$) in one place, and from the other codewords in more than one place. Hence the *most likely* word sent was $v$, so it makes sense to decode the received word as $v$. This is an example of error *correction*.

**Exercise 1.18** Consider the code $Rep(3)$ from Example 1.17. If the word 100010 is received, what is the most likely codeword to have been sent?

The decoding method used in the previous example makes intuitive sense as a decoding mechanism: correct any received word to the codeword which differs from the received word in as few bit places as possible. We can formally prove that this is valid. Recall that $\Phi_p(v, w)$ is the probability that the word $w$ is received if the codeword $v$ is transmitted over a BSC with reliability $p$. In practice, we know the word received $w$, but we do not know the codeword transmitted $v$. However, we know all of the codewords, so can calculate $\Phi_p(u, w)$ for each codeword $u \in C$. Clearly, we want to choose the *most-likely* transmitted word, which means we choose the codeword $v$ for which

$$\Phi_p(v, w) = \max\{\Phi_p(u, w) | u \in C\}.$$

We can choose such a codeword via the following theorem:

**Theorem 1.19** *Suppose communication is via a BSC with reliability $p$, $0.5 < p < 1$. Let $v_1$ and $v_2$ be codewords of length $n$, and $w$ a word of length $n$. Suppose that $v_1$ and $w$ disagree in $d_1$ positions, and that $v_2$ and $w$ disagree in $d_2$ positions. Then*

$$\Phi_p(v_1, w) \leq \Phi_p(v_2, w) \text{ if and only if } d_1 \geq d_2.$$

**Proof:** We have

$$\begin{array}{rrcl} & \Phi_p(v_1, w) & \leq & \Phi_p(v_2, w) \\ \leftrightarrow & p^{n-d_1}(1-p)^{d_1} & \leq & p^{n-d_2}(1-p)^{d_2} \\ \leftrightarrow & \left(\frac{p}{1-p}\right)^{d_2-d_1} & \leq & 1 \\ \leftrightarrow & d_2 & \leq & d_1 \end{array}$$

(since $\frac{p}{1-p} > 1$). □

Thus, as we wish to maximise $\Phi_p(u, w)$, we correct the received word $w$ to the codeword $v$ which differs from $w$ in as few positions as possible.

**Exercise 1.20** Suppose that $w = 0010110$ is received over a BSC with $p = 0.9$. Which of the following codewords is most likely to have been sent?

$$1001011, 1111100, 0001110, 0011001, 1101001.$$

What would have been the case if the channel instead had reliability $p = 0.51$?

## 1.6 Weights and distances

We need an efficient way of finding which codeword is closest to a received word. If there are many codewords, it's not practical to check every received word against every possible codeword. (For example, the code used on the Voyager Mission had $2^{12} = 4096$ codewords.)

Recall that $K^n$ consists of all the binary vectors (words) of length $n$.

**Exercise 1.21** If $u, v, w$ are words in $K^n$, show that

1. $v + w = \mathbf{0}$ iff $v = w$;
2. if $v$ is transmitted over a BSC and $w$ is received, then $u = v + w$ will be a word containing a 1 in exactly those places in which $v$ and $w$ differ.

**Definition 1.22** Given two words $v$ and $w$ in $K^n$, the corresponding *error pattern* or *error* is defined by $u = v + w$, and is 1 in exactly those places in which $v$ and $w$ differ.

**Definition 1.23** Let $v \in K^n$ be a word of length $n$. Then the *weight* or *Hamming weight* of $v$ is the number of occurrences of the digit 1 in $v$. We denote the weight of a word $v$ as $wt(v)$.

**Definition 1.24** The *Hamming distance* $d(u, v)$ between two words $u, v \in K^n$ is the number of places in which their bits differ.

**Exercise 1.25** Let $x = 111001$, $y = 001111$ and $z = 101010$. Find $wt(x)$, $d(x, y)$ and $d(y, z)$.

Hamming distance satisfies the properties of a *metric*. That is, if $x, y, z \in K^n$, then

1. $d(x, y) \geq 0$
2. $d(x, y) = 0$ iff $x = y$
3. $d(x, y) = d(y, x)$ (symmetry)
4. $d(x, z) \leq d(x, y) + d(y, z)$ (triangle inequality)

Note that if $v$ and $w$ are codewords and $u$ is the error pattern $u = v + w$, then we have $d(v, w) = wt(u)$. That is, $d(v, w) = wt(v + w)$.

If $u = v + w$, then the probability formula from Theorem 1.19 can be rewritten as

$$\Phi_p(v, w) = p^{n-wt(u)}(1 - p)^{wt(u)}.$$

We refer to $\Phi_p(v, w)$ as the *probability of the error pattern* $u = v + w$.

## 1.7 Maximum Likelihood Decoding

Now we are ready to describe more formally how decoding is done in general. There are two commonly used approaches to decoding.

**Definition 1.26** *Complete Maximum Likelihood Decoding* or *CMLD*
Let $C$ be the set of codewords, $v \in C$ be the codeword transmitted, and $w$ the word received. If there is a unique codeword $v \in C$ for which $d(v, w) < d(v_1, w)$ for all $v_1 \in C, v_1 \neq v$, then we decode $w$ as $v$. If there are two or more codewords which are all closest to $w$, then we arbitrarily select one of the equally closest codewords and decode $w$ as that arbitrary choice.

**Definition 1.27** *Incomplete Maximum Likelihood Decoding* or *IMLD*
Again, if there is a unique word $v$ in $C$ closest to $w$ then we decode $w$ as $v$. However, if there are two or more codewords which are all equally close to $w$, then we request retransmission.

Unless stated otherwise, we will always assume that IMLD is being used.

Note that with Maximum Likelihood Decoding, we are always going to select the codeword which is *closest* to the received word. This is not necessarily the same as selecting the *correct* or *transmitted*

codeword: it may be the case that so many errors occurred in transmission that the codeword closest to the received word is not the same as the codeword which was transmitted.

One of our aims is to ensure that decoding to the *closest* codeword will (almost) always produce the *correct* codeword.

Using IMLD, the codeword $v_1 \in C$ closest to the received word $w$ is the codeword for which $d(v_1, w)$ is least. By Theorem 1.19, this codeword $v_1$ has the largest probability $\Phi_p(v_1, w)$ of being the codeword transmitted. Since $d(v_1, w) = wt(v_1 + w)$, we can restate the result of Theorem 1.19 as

$$\Phi_p(v_1, w) \leq \Phi_p(v_2, w) \text{ iff } wt(v_1 + w) \geq wt(v_2 + w).$$

That is, *the most likely codeword sent is the one with the error pattern of smallest weight.*

Thus given a received word $w$, the decoding strategy for IMLD is to examine the error patterns $u = v + w$ for all codewords $v \in C$ and decode $w$ as the codeword $v^*$ which gives the error pattern $u^*$ of smallest weight.

**Example 1.28** If $C = \{0000, 1010, 0111\}$, construct an IMLD table showing, for each possible received word $w$, to which codeword $w$ will be decoded. Remember that retransmission will be requested in the event that there is no unique codeword which minimises the weight of the error pattern.

| received word | error patterns | | | most likely codeword |
|---|---|---|---|---|
| $w$ | $0000 + w$ | $1010 + w$ | $0111 + w$ | $v$ |
| 0000 | 0000 | 1010 | 0111 | 0000 |
| 0001 | 0001 | 1011 | 0110 | 0000 |
| 0010 | 0010 | 1000 | 0101 | $--$ |
| 0011 | 0011 | 1001 | 0100 | 0111 |
| 0100 | 0100 | 1110 | 0011 | 0000 |
| 0101 | 0101 | 1111 | 0010 | 0111 |
| 0110 | 0110 | 1100 | 0001 | 0111 |
| 0111 | 0111 | 1101 | 0000 | 0111 |
| 1000 | 1000 | 0010 | 1111 | $--$ |
| 1001 | 1001 | 0011 | 1110 | $--$ |
| 1010 | 1010 | 0000 | 1101 | 1010 |
| 1011 | 1011 | 0001 | 1100 | 1010 |
| 1100 | 1100 | 0110 | 1011 | $--$ |
| 1101 | 1101 | 0111 | 1010 | 0111 |
| 1110 | 1110 | 0100 | 1001 | 1010 |
| 1111 | 1111 | 0101 | 1000 | 0111 |

When establishing a BSC and a code, it is necessary to choose the value of $n$ (length of the codewords) and the actual codewords $C$. Some choices of $n$ and $C$ are better than others. There are many criteria that are used in the choices of $n$ and $C$. For now we would like to choose codewords $C$ such that IMLD works reasonably well. To do that we need to determine, for each codeword $v$, the probability that IMLD will correctly conclude that $v$ was transmitted.

Given $n$ and $C$, we can calculate the probability $\Theta_p(C, v)$ that if $v \in C$ is transmitted over a BSC with reliability $p$ then IMLD correctly concludes that $v$ was sent.

**Definition 1.29** *Calculating the reliability of IMLD*
Assume that a codeword $v \in C$ is transmitted over a BSC with reliability $p$. Let

$$L(v) = \{x \mid x \in K^n \text{ and } x \text{ is closer to } v \text{ than to any other codeword in } C\}.$$

Then $\Theta_p(C, v)$ is the sum of all the probabilities $\Phi_p(v, w)$ as $w$ ranges over $L(v)$. That is,

$$\Theta_p(C, v) = \sum_{w \in L(v)} \Phi_p(v, w).$$

$L(v)$ is precisely the set of words in $K^n$ for which, if received, IMLD will conclude that $v$ was sent. We can find $L(v)$ from an IMLD table, such as the one given above, by comparing the received words with the most likely codewords.
Note that the definition of $\Theta_p$ ignores the effect of retransmission when decoding is ambiguous (so the received word might be correctly decoded on second transmission), but $\Theta_p$ is still a reasonable lower bound on the probability of correct decoding.

**Example 1.30** Suppose $p = 0.9$, $n = 4$ and $C = \{0000, 1010, 0111\}$ (as in Example 1.28). Compute $\Theta_p(C, 0000)$.

For $v = 0000$, $L(v) = \{0000, 0100, 0001\}$. Thus

$$\begin{aligned} \Theta_p(C, 0000) &= \Phi_p(0000, 0000) + \Phi_p(0000, 0100) + \Phi_p(0000, 0001) \\ &= p^4 + p^3(1-p) + p^3(1-p) \\ &= 0.8019 \end{aligned}$$

**Exercise 1.31** Let $p = 0.9$, $n = 4$ and $C = \{0000, 1010, 0111\}$ (as in Examples 1.28 and 1.30. Compute $\Theta_p(C, 1010)$ and $\Theta_p(C, 0111)$.

We can see that for $C = \{0000, 1010, 0111\}$, IMLD works reasonably well if 0111 is transmitted, but performs poorly if 0000 or 1010 is transmitted. Thus we see that $C$ is not a very good choice as a code.

## 1.8 Error detection and correction

Now we can formalise the definition of error detection.

**Definition 1.32** We say that a code $C$ *detects* the error pattern $u$ if and only if $v+u$ is not a codeword, for every $v \in C$.

**Exercise 1.33** Let $C = \{001, 101, 110\}$. Show that $C$ detects the error pattern $u_1 = 010$, but does not detect the error pattern $u_2 = 100$.

A good method of determining which error patterns a code can detect is to first determine which error patterns it *cannot* detect. Given a code $C$ and any pair of codewords $v$ and $w$, if $e = v + w$ then $C$ cannot detect the error pattern $e$ (as $v + e = w$ which is a codeword). Thus the set of all error patterns which cannot be detected by $C$ is the set of all words that can be written as the sum of two codewords.

**Exercise 1.34** Find all error patterns which can be detected by $C = \{1000, 0100, 1111\}$.

For certain codes, we can calculate some of the error patterns which the code can detect, without needing to go through any of the above calculations. We make use of the concept of the *distance* between two codewords, defined in Definition 1.24, and define the concept of the *distance of a code*.

**Definition 1.35** The *distance* (or *minimum distance*) of a code is the smallest distance between any pair of distinct codewords. That is, we define $\delta$ to be the *distance* of a code $C$ if

$$\delta = \min_{v,w \in C, v \neq w} d(v, w).$$

We know that $d(v, w) = wt(v + w)$, so $\delta$ is the smallest value of $wt(v + w)$ as $v, w$, $v \neq w$ range over all possible pairs of codewords.

An $(n, k)$ code with minimum distance $\delta$ will sometimes be written as a $(n, k, \delta)$ code.

**Exercise 1.36** Find the distance of the code $C = \{0000, 1010, 0111\}$.

**Exercise 1.37** Find the distance of an $n$-fold repetition code.

We now have a very important theorem, which makes direct use of the definition of distance given above.

**Theorem 1.38** *Let $C$ be a code with minimum distance $\delta$. Then $C$ will detect all non-zero error patterns of weight less than or equal to $\delta - 1$. Moreover, there is at least one error pattern of weight $\delta$ which $C$ will not detect.*

**Proof:** Let $u$ be a non-zero error pattern with $wt(u) \leq \delta - 1$, and let $v \in C$. Then $d(v, v + u) = wt(v + (v + u)) = wt(u) < \delta$. Since $C$ has distance $\delta$, $v + u \notin C$. Therefore $C$ detects $u$.
From the definition of $\delta$, there exist codewords $v, w \in C$ with $d(v, w) = \delta$. Consider the error pattern $u = v + w$. Now $w = v + u \in C$, so $C$ will not detect the error pattern $u$ of weight $\delta$. □

Note that a code $C$ with minimum distance $\delta$ may possibly detect *some* error patterns of weight $\delta$ or more, but it does not detect *all* error patterns of weight $\delta$.

**Exercise 1.39** Show that the code defined in Exercise 1.36 detects the error pattern 1111 of weight 4, but find an error pattern of weight 2 which $C$ does not detect.

**Definition 1.40** A code $C$ is said to be $x$ *error detecting* if it detects **all** error patterns of weight at most $x$, and does not detect at least one error pattern of weight $x + 1$. By Theorem 1.38, if $C$ has distance $\delta$ then $C$ is $\delta - 1$ error detecting.

Equivalently, for a code $C$ to be $e$ error detecting, it must have distance $e + 1$.

**Exercise 1.41** Let $C = \{0000, 1010, 0111\}$.

1. What is the distance of $C$?
2. $C$ is $x$-error detecting. What is $x$?
3. Find all error patterns that $C$ does detect, and hence show that 1010 is the only error pattern of weight 2 that $C$ does not detect.

Now we can formalise the definition of error correction.

**Definition 1.42** A code $C$ *corrects an error pattern* $u$ if, for all $v \in C$, $v + u$ is closer to $v$ than to any other word in $C$.

This is equivalent to saying that $C$ corrects the error pattern $u$ if adding $u$ to any codeword $v$ results in a word which is still closer to $v$ than to any other codeword.

**Definition 1.43** A code $C$ is said to be $x$ *error correcting* if it corrects all error patterns of weight at most $x$, and does not correct at least one error pattern of weight $x + 1$.

Given a codeword $v$ we can think of a "sphere" in $n$ dimensions of radius $x$ centred on the codeword $v$ by saying that another word $w$ falls within this sphere iff $w$ is within distance $x$ of codeword $v$. Intuitively, $C$ is $x$ error correcting if it possible to take each codeword in $C$, draw a "sphere of radius $x$" over each codeword, and have no two spheres intersect. Then any received word which falls within the sphere of a codeword will be corrected unambiguously to the codeword on which the sphere is based.

**Exercise 1.44** If $C$ is a code and $v = 0000 \in C$, list all words within a sphere of radius 2 of $v$.

**Exercise 1.45** Let $C = \{000, 111\}$. Show that $C$ corrects any error pattern of weight one, but does not correct any error pattern of weight two.

The process of IMLD picks the "most-likely" codeword. When we relate error correction and the distance of a code, we need to ensure that the most-likely codeword is the correct codeword, and not just the closest codeword.

Let $v_1, v_2$ be codewords with $d(v_1, v_2) = \delta$. Clearly, if $v_1$ is transmitted and errors occur in $\delta - 1$ of the places in which $v_1$ and $v_2$ differ, the received word will be the same as if $v_2$ had been transmitted and a single error had occurred in the other place in which $v_1$ and $v_2$ differ. Thus $\delta - 1$ errors in $v_1$ can be equivalent to 1 error in $v_2$. Similarly, $\delta - 2$ errors in $v_1$ can give the same received word as 2 errors in $v_2$, and so on.

Suppose that we are using a code $C$ with distance $\delta$ and a codeword $v$ is transmitted and the word $w$ is received where $d(v, w) = \delta - 1$. Then we can *detect* that up to $\delta - 1$ errors have occured. However, care must be taken with error correction since $\delta - 1$ errors in $v$ may be indistinguishable from just 1 error in a different codeword and so in choosing the closest codeword to the received word $w$, the process of IMLD may return the incorrect codeword.

**Example 1.46** Let $C = \{000, 111\}$, so $C$ has distance 3 and hence can detect all error patterns of weight at most 2. If the word $w = 010$ is received, it is more likely (and error correction will assume) that the error pattern which occurred was 010 (of weight 1, with transmitted word 000), rather than an error pattern 101 (of weight 2, with transmitted word 111).

We now state a fundamental theorem which forms the basis for error correcting codes.

**Theorem 1.47** *Let $C$ be a code with minimum distance $\delta$.*

*If $\delta$ is odd, then $C$ can correct all error patterns with weight less than or equal to* $\dfrac{\delta - 1}{2}$.

*If $\delta$ is even, then $C$ can correct all error patterns with weight less than or equal to* $\dfrac{\delta - 2}{2}$.

Theorem 1.47 can be justified intuitively. If $C$ has distance $\delta$, then any two codewords are at least distance $\delta$ apart. If $\delta$ is odd and up to $(\delta - 1)/2$ errors occur, or if $\delta$ is even and up to $\delta/2$ errors occur, then the received word will not be a codeword, so it is clearly possible to *detect* that this number of errors has occurred. In most of these cases, it is also possible to *correct* the error, by selecting the unique codeword which is closest to the received word. However, if $\delta$ is even and $\delta/2$ errors have occurred, then it is possible for the received word to be equidistant from two distinct codewords. Thus it may not be possible to unambiguously select the *closest* codeword, so error correction may not work.

Using $\lfloor x \rfloor$ to denote the integer part of $x$, we see that a code $C$ of distance $\delta$ can correct up to $\left\lfloor \dfrac{\delta - 1}{2} \right\rfloor$ errors. We note that there is at least one error pattern of weight $1 + \left\lfloor \dfrac{\delta - 1}{2} \right\rfloor$ which $C$ does not correct. A code of distance $\delta$ *may* correct *some* error patterns of weight larger than that specified in Theorem 1.47. However, it will not correct *all* such error patterns.

**Exercise 1.48** Assume that the information to be transmitted consists of all possible strings of length 3. Label the message bits $x_1$, $x_2$ and $x_3$, and let $C$ contain codewords of length $n = 6$ formed by appending three extra digits $x_4, x_5$ and $x_6$, so that each of the following sums are even:

$$x_2 + x_3 + x_4,\ x_1 + x_3 + x_5,\ x_1 + x_2 + x_6.$$

1. List the codewords of $C$.
2. What is the distance of $C$?
3. How many errors can $C$ detect and correct?
4. What is the information rate of $C$?

**Exercise 1.49** Repeat the previous example, but instead form a code $D$ by repeating each of the three message bits three times.

**Exercise 1.50** Compare the rates of codes $C$ and $D$ from the two previous examples.

Later, we will see how to construct a family of codes called the *Hamming codes*. We will see that the Hamming code of length 7 is 2 error detecting and 1 error correcting, but has information rate 4/7.

# 2 Linear codes I

The essential goal of coding theory is to find codes which transmit information at reasonable rates, yet also detect and correct most transmission errors. In this section we discuss a broad class of codes which provide these features, based heavily on linear algebra.

## 2.1 Introduction to linear codes

**Definition 2.1** A **linear code** is a code in which the sum (mod 2) of any two codewords is also a codeword. That is, $C$ is linear iff for any pair of codewords $v, w \in C$, we also have $v + w \in C$.

Almost every code we consider for the remainder of this course will be linear.

**Exercise 2.2** Show that $C_1 = \{0000, 0101, 1010, 1111\}$ is linear, but that $C_2 = \{0000, 1001, 1010, 0011, 1111\}$ is not linear.

**Exercise 2.3** Explain why any linear code must contain the zero word.

**Exercise 2.4** Five of the eight codewords of a linear code are

$$0001111, 0110101, 1010011, 1011100, 1100110.$$

Find the remaining three codewords.

One of the advantages of a linear code is that its distance is easy to find.

**Theorem 2.5** *For any linear code $C$, the distance of $C$ is the weight of the nonzero codeword of smallest weight.*

**Proof:** Let $C$ be a code of distance $\delta$, and let $w$ be the nonzero codeword of smallest weight. Certainly, $\delta \leq wt(w)$. Assume that there are two codewords $v_1, v_2 \in C$ such that $\delta = d(v_1, v_2) = d < wt(w)$. As $C$ is linear, $v = v_1 + v_2$ must be a codeword, of weight $d(v_1, v_2) < wt(w)$. But this contradicts the assumption that $w$ is the nonzero codeword of smallest weight. □

**Exercise 2.6** Find the distance of the linear code $C = \{0000, 1100, 0011, 1111\}$.

It is easy to find the distance of a linear code. Other advantages of linear codes include:

1. For linear codes, there is a procedure for IMLD which is simpler and faster than we have seen so far (for some linear codes, there are very simple decoding algorithms).
2. Encoding using a linear code is faster and requires less storage space than for arbitrary non-linear codes.
3. The probabilities $\Theta_p(C, v)$ are straightforward to calculate for a linear code.
4. It is easy to describe the set of error patterns that a linear code will detect.
5. It is much easier to describe the set of error patterns a linear code will correct than it is for arbitrary non-linear codes.

Since a subset $U \subseteq K^n$ is a subspace of $K^n$ iff $U$ is closed under addition, we conclude that $C$ is a linear code iff $C$ is subspace of $K^n$.

Thus, for any subset $S$ of $K^n$, the span of $S$ is a linear code, $C = \langle S \rangle$.

The *dimension of a linear code* is the dimension of the corresponding subspace of $K^n$. Similarly, a *basis for a linear code* is a basis for the corresponding subspace of $K^n$.

If a linear code $C$ has dimension $k$ and if $B = \{v_1, v_2, \ldots, v_k\}$ is a basis for $C$, then each codeword $w$ in $C$ can be written as

$$w = \alpha_1 v_1 + \alpha_2 v_2 + \ldots + \alpha_k v_k$$

for a unique choice of digits $\alpha_1, \alpha_2, \ldots, \alpha_k$. Noting that each $\alpha_i$ is 0 or 1, for $1 \leq i \leq k$, there are $2^k$ distinct choices for $\alpha_1, \alpha_2, \ldots, \alpha_k$.

We thus have the following very important theorem:

**Theorem 2.7** *A linear code of dimension $k$ contains precisely $2^k$ codewords.*

Thus, using the notation introduced earlier in the course, a linear code with length $n$, dimension $k$ and distance $\delta$ is an $(n, k)$ linear code of distance $\delta$, or equivalently, an $(n, k, \delta)$ linear code. Such a code has information rate $k/n$.

## 2.2 Dual codes

We now see how to derive a new code from a given linear code, using the orthogonal complement.

**Definition 2.8** For $S \subseteq K^n$, if $C = \langle S \rangle$, then we write $C^\perp = S^\perp$ and call $C^\perp$ the *dual code* of $C$.

**Theorem 2.9** *Let $C = \langle S \rangle$ be the linear code generated by a subset $S$ of $K^n$. If the dimension of $C$ is $k_1$ and the dimension of $C^\perp$ is $k_2$ then we must have $k_1 + k_2 = n$.*

**Exercise 2.10** Suppose that $C$ is a $(9, 4)$ linear code. How many codewords are in $C$? How many codewords are in $C^\perp$?

## 2.3 Bases for codes

In this section we develop methods for finding bases for a linear code $C = \langle S \rangle$ and its dual $C^{\perp}$.

**Algorithm 2.11 Algorithm for finding a basis for $C = \langle S \rangle$.**
Let $S$ be a nonempty subset of $K^n$. Form the matrix $\mathbf{A}$ whose **rows** are the words in $S$. Use EROS to find a REF of $\mathbf{A}$. Then the nonzero rows of the REF of $\mathbf{A}$ form a basis for $C = \langle S \rangle$.

Algorithm 2.11 works because the rows of $\mathbf{A}$ generate $C$, and EROS simply interchange codewords (rows) or replace one codeword (row) with the sum of two rows (another codeword) giving a new set of codewords which still generates $C$. Clearly the nonzero rows in a matrix in REF are linearly independent. Note that Algorithm 2.11 does not produce a unique basis for $C = \langle S \rangle$, and there is no guarantee that the words in the basis occur in the given set $S$.

**Exercise 2.12** Use Algorithm 2.11 to find a basis for the linear code $C = \langle S \rangle$ where $S = \{11101, 10110, 01011, 11010\}$.

Now we give an algorithm for finding a basis for the dual code $C^{\perp}$. This algorithm incorporates Algorithm 2.11, so it also gives a basis for $C$.

**Algorithm 2.13 Algorithm for finding bases for $C$ and $C^{\perp}$.**
Let $S$ be a nonempty subset of $K^n$. Form the matrix $\mathbf{A}$ whose rows are the words in $S$. Use EROS to place $\mathbf{A}$ in RREF. Let $\mathbf{G}$ be the $k \times n$ matrix consisting of all the nonzero rows of the RREF. Then the rows of $\mathbf{G}$ form a basis for $C$. Let $\mathbf{X}$ be the $k \times (n-k)$ matrix obtained from $\mathbf{G}$ by deleting the leading columns of $\mathbf{G}$. Form an $n \times (n-k)$ matrix $\mathbf{H}$ as follows:

1. in the rows of $\mathbf{H}$ corresponding to the leading columns of $\mathbf{G}$, place, in order, the rows of $\mathbf{X}$
2. in the remaining $n-k$ rows of $\mathbf{H}$, place, in order, the rows of the $(n-k) \times (n-k)$ identity matrix $\mathbf{I}_{n-k}$.

Then the columns of $\mathbf{H}$ form a basis for $C^{\perp}$.

Here is a more intuitive description of Algorithm 2.13. Start with matrix $\mathbf{A}$, and use EROS to convert:

$$\mathbf{A} \to \begin{pmatrix} \mathbf{G} \\ \mathbf{0} \end{pmatrix} \text{ in RREF.}$$

Then permute the columns of $\mathbf{G}$ to form $\mathbf{G} \to \mathbf{G}' = (\mathbf{I}_k \ \mathbf{X})$.
Form a matrix $\mathbf{H}'$ as follows:

$$\mathbf{H}' = \begin{pmatrix} \mathbf{X} \\ \mathbf{I}_{n-k} \end{pmatrix}.$$

Apply the inverse of the permutation applied to the columns of $\mathbf{G}$ to the rows of $\mathbf{H}'$ to form $\mathbf{H}$.

**Example 2.14** Consider the code $C = \langle S \rangle$ where $S = \{11010, 10001, 01001, 11000\}$. Use Algorithm 2.13 to find a basis for $C$ and a basis for $C^\perp$.

$$\mathbf{A} = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$\to \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \to \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \quad \text{which is in RREF}$$

A basis for $C$ is $\{10001, 01001, 00010\}$.

Now we have $\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$, so we have $\mathbf{G}' = \left(\begin{array}{ccc|cc} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \end{array}\right) = \begin{pmatrix} \mathbf{I}_3 & \mathbf{X} \end{pmatrix}$.

Thus $k = 3$ and $\mathbf{X} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 0 \end{pmatrix}$.

The rows of $\mathbf{X}$ are placed in the first three rows, respectively, of the $5 \times (5-3)$ matrix $\mathbf{H}'$. The remaining rows of $\mathbf{H}'$ are filled with the the $2 \times 2$ identity matrix. Thus

$$\mathbf{H}' = \left(\begin{array}{cc} 0 & 1 \\ 0 & 1 \\ 0 & 0 \\ \hline 1 & 0 \\ 0 & 1 \end{array}\right), \quad \text{and so} \quad \mathbf{H} = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{pmatrix}.$$

Thus a basis for $C^\perp$ is $\{00100, 11001\}$.

**Exercise 2.15** Verify that for the previous example, $\mathbf{GH} = \mathbf{0}$.

We make a few comments to justify why Algorithm 2.13 works. The $n-k$ columns of $\mathbf{H}$ are linearly independent and $\dim C^\perp = n - \dim C = n-k$, so the columns of $\mathbf{H}$ are a basis for a subspace of the correct dimension. Furthermore,

$$\mathbf{G}'\mathbf{H}' = (\mathbf{I}_k \;\; \mathbf{X}) \begin{pmatrix} \mathbf{X} \\ \mathbf{I}_{n-k} \end{pmatrix} = \mathbf{X} + \mathbf{X} = \mathbf{0}.$$

We apply the same permutation to the columns of $\mathbf{G}'$ and to the rows of $\mathbf{H}'$ to obtain $\mathbf{G}$ and $\mathbf{H}$, so we still get $\mathbf{GH} = \mathbf{0}$. Thus each row of $\mathbf{G}$ is orthogonal to each column of $\mathbf{H}$, and so if the rows of $\mathbf{G}$ form a basis for $C$ then the columns of $\mathbf{H}$ must form a basis for $C^\perp$.

**Exercise 2.16** If $S = \{101010, 010101, 111111, 000111, 101100\}$, find a basis $B$ for the code $C = \langle S \rangle$, and find a basis $B^\perp$ for the dual code $C^\perp$. Determine the number of codewords in each of $C$ and $C^\perp$.

## 2.4 Generating matrices

We now use the material from the previous section to create a matrix which is used in the encoding process for a linear code.

Recall that the *rank* of a matrix $\mathbf{M}$ over $K$ is the number of nonzero rows in any REF of $\mathbf{M}$.

**Definition 2.17** If $C$ is a linear code of length $n$ and dimension $k$, then any matrix whose rows form a basis for $C$ is called a *generating matrix* for $C$. Such matrices must have $k$ rows and $n$ columns, and have rank $k$.

Then we have the following two theorems:

**Theorem 2.18** *A matrix* $\mathbf{G}$ *is a generating matrix for some linear code* $C$ *iff the rows of* $\mathbf{G}$ *are linearly independent (so the rank of* $\mathbf{G}$ *equals the number of rows of* $\mathbf{G}$*).*

**Theorem 2.19** *If* $\mathbf{G}$ *is a generating matrix for a linear code* $C$*, then any matrix row equivalent to* $\mathbf{G}$ *is also a generating matrix for* $C$*. In particular, any linear code has a generating matrix in RREF.*

To find a generating matrix for a linear code $C$, we can form the matrix whose rows are the nonzero codewords in $C$. By the definition of 'linear' we must have $C = \langle C \rangle$, so we can use Algorithm 2.11 to produce a basis for $C$. Then the matrix whose rows are these basis vectors forms a generating matrix for $C$.

**Exercise 2.20** Find a generating matrix in RREF for the linear code $C = \{0000, 1110, 0111, 1001\}$.

A $k \times n$ matrix $\mathbf{G}$ is a generating matrix for an $(n, k)$ linear code if the binary words that can be expressed as the sum of a subset of the rows of $\mathbf{G}$ are exactly the codewords of $C$.

**Exercise 2.21** Let $C = \{000000, 001110, 010101, 011011, 100011, 101101, 110110, 111000\}$. Show that

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

is a generating matrix for $C$.

Having defined a generating matrix, we can use standard techniques from linear algebra to show how messages of length $k$ are encoded as codewords of length $n$.

Let $C$ be a linear $(n, k)$ code. If $\mathbf{G}$ is a generating matrix for $C$ and if $u$ is a word of length $k$ written as a row vector, then $v = u\mathbf{G}$ is a codeword in $C$, since $v$ is a linear combination of the rows of $\mathbf{G}$, and these rows form a basis for $C$.

Let a message be $u = \alpha_1\alpha_2 \ldots \alpha_k$, and let $g_1, g_2, \ldots, g_k$ be the rows of $\mathbf{G}$. Then

$$v = u\mathbf{G} = \alpha_1 g_1 + \alpha_2 g_2 + \ldots + \alpha_k g_k$$

is a codeword in $C$.

It is easy to see that if $u_1\mathbf{G} = u_2\mathbf{G}$ then we must have $u_1 = u_2$, since each codeword in $C$ is a *unique* linear combination of the codewords in a basis. Thus no codeword $v = u\mathbf{G}$ in $C$ is produced by more than one word $u \in K^k$.

These observations lead to an important theorem.

**Theorem 2.22** *If* $\mathbf{G}$ *is a generating matrix for a linear code* $C$ *of length* $n$ *and dimension* $k$ *then* $v = u\mathbf{G}$ *ranges over all* $2^k$ *codewords in* $C$ *as* $u$ *ranges over all* $2^k$ *words of length* $k$*. Thus* $C$ *is the set of all words* $u\mathbf{G}$*,* $u \in K^k$*. Moreover,* $u_1\mathbf{G} = u_2\mathbf{G}$ *iff* $u_1 = u_2$*.*

If our messages contain $k$ bits, we can use Theorem 2.22 to encode the messages. Indeed, that theorem says that the messages which can be encoded by a linear $(n, k, \delta)$ code are exactly all messages $u \in K^k$. The $k$-bit message is encoded as an $n$-bit codeword, so only $k$ digits in any codeword are used to carry the message.

**Exercise 2.23** Let $C$ be the $(5, 3)$ linear code with generating matrix

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

Assign letters to the words in $K^3$ as follows:

| 000 | 100 | 010 | 001 | 110 | 101 | 011 | 111 |
|---|---|---|---|---|---|---|---|
| A | B | E | H | M | R | T | W |

Using the generating matrix $\mathbf{G}$, encode the message BETHERE.

## 2.5 Equivalent codes

**Definition 2.24** Any $k \times n$ matrix $\mathbf{G}$ with $k < n$ whose first $k$ columns form the $k \times k$ identity matrix $\mathbf{I}_k$, so $\mathbf{G} = (\mathbf{I}_k \ \mathbf{X})$, is said to be in *standard form*. We note that $\mathbf{G}$ clearly must automatically have linearly independent rows and is in RREF. Thus $\mathbf{G}$ is a generating matrix for some linear code of length $n$ and dimension $k$. We say that the code $C$ generated by $\mathbf{G}$ is *systematic*.

Not all linear codes have a generating matrix in standard form. For example, the code $C = \{000, 100, 001, 101\}$ has six possible generating matrices. Its generating matrix in RREF is:

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

Hence $C$ is not a systematic code.

**Exercise 2.25** Is the linear code $C = \{00000, 10110, 10101, 00011\}$ a systematic code?

There are several important advantages in using a linear code with a generating matrix in standard form. Let $C$ be a linear $(n, k)$ code with generating matrix $\mathbf{G}$ in standard form.

1. Since $\mathbf{G}$ is in standard form, $\mathbf{G} = (\mathbf{I}_k \ \mathbf{X})$, and so Algorithm 2.13 immediately gives a basis for the dual code $C^\perp$ (without any need to permute columns of $\mathbf{G}$ and rows of $\mathbf{H}'$) as the columns of
$$\mathbf{H} = \begin{pmatrix} \mathbf{X} \\ \mathbf{I}_{n-k} \end{pmatrix}.$$

2. By Theorem 2.22, each codeword $v \in C$ can be written as $u\mathbf{G}$ for precisely one word $u \in K^k$. We can think of $u$ as the message to be sent, but we transmit the (longer) codeword $v = u\mathbf{G}$. Errors may occur during transmission, and IMLD aims to recover $v$ from the received word. Assuming $v$ is correctly inferred, the receiver now needs to recover the message $u$ from $v$. Since $\mathbf{G}$ is in standard form, it is very easy to recover $u$ from $v$. The codeword $v$ is given by:
$$v = u\mathbf{G} = u\,(\mathbf{I}_k \ \mathbf{X}) = (u\mathbf{I}_k \ u\mathbf{X}) = (u \ u\mathbf{X}).$$
Thus, the message bits $u$ form the first $k$ bits of the codeword $v$.

We have the following important theorem:

**Theorem 2.26** *If $C$ is a linear code of length $n$ and dimension $k$ with generating matrix $\mathbf{G}$ in standard form and a message $u$ is to be transmitted, then the first $k$ digits in the codeword $v = u\mathbf{G}$ form the message word $u \in K^k$.*

**Exercise 2.27** Suppose $C$ is a $(7,4)$ linear code with generating matrix

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{I}_4 & \mathbf{X} \end{pmatrix}.$$

Calculate the codeword corresponding to the message word 0111. Which message word corresponds to the codeword 1011000?

We have just seen several advantages of using a linear code with generating matrix in standard form. However, we also saw that some codes have no generating matrix in standard form. Given such a code, can we produce a different code, which is somehow equivalent to our original code, but with a generating matrix in standard form?

**Definition 2.28** If $C$ is any block code of length $n$, we can obtain a new block code $C'$ of length $n$ by choosing a particular permutation of the $n$ digits, and then consistently rearranging every word in $C$ according to that permutation. The resulting code $C'$ is said to be *equivalent* to $C$.

**Example 2.29** Let $n = 5$ and $C = \{11111, 01111, 00111, 00011, 00001\}$. If we rearrange the digits into the order $2, 1, 4, 5, 3$ then the resulting code $C' = \{11111, 10111, 00111, 00110, 00010\}$ is equivalent to $C$.

Rearranging digits in codewords is equivalent to rearranging columns in a generating matrix. Hence we have the following theorem:

**Theorem 2.30** *Any linear code $C$ is equivalent to a linear code having a generating matrix in standard form.*

**Proof:** If $\mathbf{G}$ is a generating matrix for $C$, place $\mathbf{G}$ in RREF. Rearrange the columns of the RREF so that the columns containing the leading ones come first and hence form an identity matrix. The result is a generating matrix $\mathbf{G}'$ in standard form for a code $C'$ which is equivalent to $C$. □

**Example 2.31** At the start of this section, we saw that the code $C$ generated by

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

has no generating matrix in standard form. However, if the order of transmission of digits is changed so that the third digit is transmitted before the second digit, then we obtain a new code $C'$ with generating matrix

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

The codes $C' = \{000, 100, 010, 110\}$ and $C = \{000, 100, 001, 101\}$ are equivalent codes.

Note that $C$ and $C'$ are *different* codes, although they have many similar properties. They are both linear, both have length 3, dimension 2 and distance 1. However, $C'$ has an important advantage: it has a generating matrix $\mathbf{G}'$ in standard form.

**Exercise 2.32** In Exercise 2.25, we showed that $C = \{00000, 10110, 10101, 00011\}$ is not systematic. Find a systematic code $C'$ which is equivalent to $C$.

## 2.6 Parity Check Matrices

We now develop another matrix for linear codes, closely related to the generating matrix, but used in error detection and correction.

**Definition 2.33** A matrix $\mathbf{H}$ is called a *parity check matrix* for a linear code $C$ if the *columns* of $\mathbf{H}$ form a basis for the dual code $C^{\perp}$.

If $C$ has length $n$ and dimension $k$ then, since the sum of the dimensions of $C$ and $C^{\perp}$ is $n$, any parity check matrix for $C$ must have $n$ rows, $n-k$ columns and rank $n-k$.

**Theorem 2.34** *A matrix* $\mathbf{H}$ *is a parity check matrix for some linear code iff the columns of* $\mathbf{H}$ *are linearly independent.*

The next theorem then describes a linear code in terms of its parity check matrix.

**Theorem 2.35** *If* $\mathbf{H}$ *is a parity check matrix for some linear code* $C$ *of length* $n$*, then* $C$ *consists precisely of all words* $v \in K^n$ *such that* $v\mathbf{H} = \mathbf{0}$*.*

Some references on coding theory write $\mathbf{H}$ as its transpose, that is, as a $(n-k) \times n$ matrix, and instead of writing $v\mathbf{H} = \mathbf{0}$, write $\mathbf{H}v^T = \mathbf{0}^T$.

**Exercise 2.36** Let $C = \{0000, 0011, 1100, 1111\}$, so $C$ is a $(4, 2)$ linear code. Verify that

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

is a parity check matrix for $C$.

We explain why the matrix $H$ is called a *parity check matrix*.

For an $(n, k)$ linear code, the matrix equation $x\mathbf{H} = \mathbf{0}$, where $x = (x_1, x_2, \ldots, x_n)$, is a shorthand way of writing a set of $n-k$ simultaneous linear equations in the variables $x_1, x_2, \ldots, x_n$. If the $j$th column of $\mathbf{H}$ has entries of 1 in the rows $m_1, m_2, \ldots m_j$, then the bit-sum

$$x_{m_1} + x_{m_2} + \ldots + x_{m_j} \equiv 0 \pmod 2.$$

In other words, the binary word $x$ satisfies $x\mathbf{H} = \mathbf{0}$ iff the sums (obtained by ordinary addition) of certain of its bits (as determined by $\mathbf{H}$) are even, and thus have even parity. The $n-k$ linear equations are thus referred to as *parity check equations* of the code, and $\mathbf{H}$ is called the *parity check matrix*.

If we are given a generating matrix $\mathbf{G}$ for a linear code $C$, we can find a parity check matrix for $C$ using Algorithm 2.13. The parity check matrix is the matrix $\mathbf{H}$ constructed in Algorithm 2.13, since the columns of $\mathbf{H}$ form a basis for $C^\perp$.

**Exercise 2.37** Find a parity check matrix for the code $C = \{000000, 101010, 010101, 111111\}$.

We can now characterise the relationship between a generating matrix and a parity check matrix for a linear code, and the relationship between these matrices for a linear code and its dual code.

**Theorem 2.38** *Matrices* $\mathbf{G}$ *and* $\mathbf{H}$ *are generating and parity check matrices respectively for some linear code* $C$ *iff*

*(i) the rows of* $\mathbf{G}$ *are linearly independent*

*(ii) the columns of* $\mathbf{H}$ *are linearly independent*

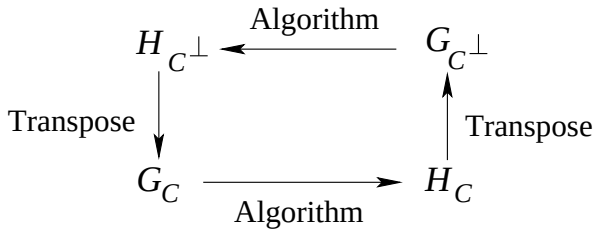
*(iii) the number of rows of* $\mathbf{G}$ *plus the number of columns of* $\mathbf{H}$ *equals the number of columns of* $\mathbf{G}$ *which equals the number of rows of* $\mathbf{H}$

*(iv)* $\mathbf{GH} = \mathbf{0}$.

**Theorem 2.39** $\mathbf{H}$ *is a parity check matrix for a linear code* $C$ *iff* $\mathbf{H}^T$ *is a generating matrix for* $C^\perp$.

Theorem 2.39 follows from Theorem 2.38 and the fact that $\mathbf{H}^T\mathbf{G}^T = (\mathbf{GH})^T = \mathbf{0}^T = \mathbf{0}$.

Given any one of the generating or parity check matrices for a code $C$ or the dual code $C^\perp$, Algorithm 2.13 and Theorem 2.39 can be used to find the other three matrices. If we let $\mathbf{G}_C$ and $\mathbf{H}_C$ denote the generating and parity check matrix respectively of $C$, and $\mathbf{G}_{C^\perp}$ and $\mathbf{H}_{C^\perp}$ denote the generating and parity check matrix respectively of $C^\perp$, then the following diagram shows how this is done.




Let's look at a comprehensive example.

**Example 2.40** Let $C$ be a linear code with parity check matrix

$$\mathbf{H}_C = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{X} \\ \mathbf{I}_2 \end{pmatrix}.$$

(a) By taking the transpose of $\mathbf{H}_C$, we find a generating matrix for $C^\perp$.

$$\mathbf{G}_{C^\perp} = \mathbf{H}_C^T = \begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}.$$

(b) Starting with $\mathbf{G}_{C^\perp}$, we use Algorithm 2.13 to find a parity check matrix for $C^\perp$. The RREF of $\mathbf{G}_{C^\perp}$ is

$$\begin{pmatrix} 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix},$$

so Algorithm 2.13 gives the following parity check matrix for $C^\perp$:

$$\mathbf{H}_{C^\perp} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

(c) We find a generating matrix for $C$ by taking the transpose of the parity check matrix $\mathbf{H}_{C^\perp}$. Thus,

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

Note that we could have also found a generating matrix for $C$ by starting with $\mathbf{H}_C$ and applying Algorithm 2.13 backwards. This would give us the following generating matrix for $C$.

$$\mathbf{G}'_C = \begin{pmatrix} \mathbf{I}_3 & \mathbf{X} \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}.$$

Now that we are familiar with the relationship between a generating matrix and a parity check matrix for a linear code $C$, we will look at how a parity check matrix can be used for error detection.

Let $C$ be a linear code with parity check matrix $\mathbf{H}$. If $v \in C$, then by definition we must have $v\mathbf{H} = \mathbf{0}$. If we receive a word $w$, we can calculate $w\mathbf{H}$. If $w\mathbf{H}$ is not equal to $\mathbf{0}$, then we have detected an error.

**Exercise 2.41** The following matrix is a parity check matrix for a linear code $C$.

$$\mathbf{H}_C = \begin{pmatrix} 1 & 1 \\ 1 & 1 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Use $\mathbf{H}_C$ to determine which of the following words are codewords of $C$.

- 11000

- 01100

- 11101

- 11010

The parity check matrix can also be used for error correction. In order to see how to do this, we first need a bit more background.

## 2.7 Distance of a linear code

We have seen that the distance of a linear code is the weight of the nonzero codeword of smallest weight. Now, we see how to find the distance of a linear code by looking at a parity check matrix of the code.

**Theorem 2.42** *Let* $\mathbf{H}$ *be a parity check matrix for a linear code* $C$*. Then* $C$ *has distance* $\delta$ *iff every set of* $\delta - 1$ *rows of* $\mathbf{H}$ *is linearly independent, while at least one set of* $\delta$ *rows of* $\mathbf{H}$ *is linearly dependent.*

**Proof:** Suppose that $C$ has distance $\delta$. Let $v \in C$ be a nonzero codeword of minimum weight. Since $C$ is linear $wt(v) = \delta$. Since $v \in C$, we have $v\mathbf{H} = \mathbf{0}$. Now $v\mathbf{H}$ is a linear combination of exactly $\delta$ rows of $\mathbf{H}$. Thus the set of $\delta$ rows of $\mathbf{H}$, corresponding to the $\delta$ nonzero bits of $v$, is a linearly dependent set. Now let $w$ be any word of weight less than or equal to $\delta - 1$. Then $w \notin C$, so $w\mathbf{H} \neq \mathbf{0}$. Thus no set of fewer than $\delta$ rows of $\mathbf{H}$ can be linearly dependent. Thus every set of $\delta - 1$ rows of $\mathbf{H}$ is linearly independent.

Now suppose that every set of $\delta - 1$ rows of $\mathbf{H}$ is linearly independent while at least one set of $\delta$ rows of $\mathbf{H}$ is linearly dependent. Then there exists a word $u$ of weight $\delta$ such that $u\mathbf{H} = \mathbf{0}$. (The word $u$ has ones in the bits corresponding to the rows in the set of $\delta$ rows of $\mathbf{H}$ that is linear depdendent.) Thus the distance of $C$ is at most $\delta$. Since every set of $\delta - 1$ rows of $\mathbf{H}$ is linearly independent, no nonzero linear combination of fewer than $\delta$ rows of $\mathbf{H}$ will give $\mathbf{0}$. Thus if $w$ is a word of weight less than $\delta$, then $w\mathbf{H} \neq \mathbf{0}$ and so $w \notin C$. Thus $u$ is a word of minimum weight in $C$ and the distance of $C$ is $\delta$. □

**Exercise 2.43** Determine the distance of the linear code $C$ with parity check matrix

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

## 2.8 Cosets

**Definition 2.44** If $C$ is a linear code of length $n$ and $u$ is any word of length $n$, we define the *coset of* $C$ *determined by* $u$ to be the set of all words of the form $v + u$ as $v$ ranges over all the codewords in $C$. We denote this coset by $C + u$. Thus

$$C + u = \{v + u \mid v \in C\}.$$

Intuitively, a coset induced by a word $u$ is the set of all possible received words if any codeword was transmitted and the error pattern $u$ occurred.

**Example 2.45** Let $C = \{000, 111\}$ and $u = 101$. Then

$$C + u = C + 101 = \{000 + 101, 111 + 101\} = \{101, 010\}.$$

Notice that if we consider the coset of $C$ induced by a codeword, then we get $C$.

$$C + 111 = \{000 + 111, 111 + 111\} = \{111, 000\} = C.$$

The following theorem gives some important results on cosets.

**Theorem 2.46** *Let $C$ be a linear code of length $n$ and let $u, v$ be words of length $n$. Then:*

*(1) The word $u$ is in the coset $C + u$.*

*(2) If $u$ is in the coset $C + v$ then $C + u = C + v$.*

*(3) If $u + v \in C$ then $u$ and $v$ are in the same coset.*

*(4) If $u + v \notin C$ then $u$ and $v$ are in different cosets.*

*(5) Every word in $K^n$ is contained in one and only one coset of $C$; thus, either $C + u = C + v$ or $C + u$ and $C + v$ have no words in common.*

*(6) $| C + u | = | C |$; that is, the number of words in a coset of $C$ is equal to the number of words in $C$.*

*(7) If $C$ has dimension $k$ then there are exactly $2^{n-k}$ different cosets of $C$, and each coset contains exactly $2^k$ words.*

*(8) The code $C$ is itself one of its cosets.*

**Example 2.47** List all of the cosets of the linear code $C = \{0000, 1011, 0101, 1110\}$.

We apply Theorem 2.46. First, $C$ is itself a coset (Theorem 2.46 (8)). Then every word in $K^4$ determines a coset (by (2) and (5)), so we can pick a word in $K^4$ not in $C$. If we take the word 1000, we have

$$C + 1000 = \{1000, 0011, 1101, 0110\}.$$

(Note that 1000 is in the coset, as suggested by (1).) Now pick another word in $K^4$ which is not in either coset so far, say 0100. Then

$$C + 0100 = \{0100, 1111, 0001, 1010\}.$$

Finally, if we choose the word 0010 not in any coset so far, we get

$$C + 0010 = \{0010, 1001, 0111, 1100\}.$$

This is all the cosets: the code $C$ has dimension $k = 2$, we have listed $2^{n-k} = 2^{4-2} = 4$ cosets each with $2^k = 2^2 = 4$ words (7), and every word in $K^4$ appears in exactly one coset (5). Also, note that for any words whose sum is in $C$, those two words are in the same coset (3), and for any two words whose sum is not in $C$, those words are in different cosets (4).

**Exercise 2.48** List the cosets of the code having the parity check matrix

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

## 2.9 MLD for Linear Codes

One of the fundamental goals of coding theory is to design codes which permit easy and rapid decoding of a received word. We describe such a method for MLD for a linear code, using the parity check matrix and cosets of the code.

Let $C$ be a linear code. Assume the codeword $v \in C$ is transmitted and the word $w$ is received, with an error pattern $u = v + w$. Clearly, $w + u = v \in C$, **so the error pattern $u$ and the received word $w$ are in the same coset of $C$** by (3) of Theorem 2.46.

Since error patterns of small weight are most likely to occur, the following is an algorithm for efficient decoding using cosets.

**Algorithm 2.49 Decoding using cosets**

If $C$ is a linear code and the word $w$ is received, we choose a word $u$ of least weight in the coset $C + w$ (which is the coset containing $w$) and conclude that $v = w + u$ was the word sent.

If there is no unique word of least weight we can either arbitrarily pick a word of least weight (CMLD), or request retransmission (IMLD).

**Example 2.50** Let $C = \{0000, 1011, 0101, 1110\}$ as in Example 2.47. Then the cosets of $C$ are:

| Coset 1 | Coset 2 | Coset 3 | Coset 4 |
|---|---|---|---|
| 0000 | 1000 | 0100 | 0010 |
| 1011 | 0011 | 1111 | 1001 |
| 0101 | 1101 | 0001 | 0111 |
| 1110 | 0110 | 1010 | 1100 |

Let $w = 1101$ be the received word. This is in the second coset in the above table, so choose $u = 1000$ as the error pattern of least weight. Hence we conclude that $v = w + u = 1101 + 1000 = 0101$ was the codeword most likely sent.

If $w = 1111$ is received, then the words 0100 and 0001 have least weight in the corresponding coset. Thus decoding either arbitrarily chooses one of these words as the most likely error pattern (CMLD), or requests retransmission (IMLD).

The hardest part of this procedure is to find the coset containing the received word $w$ and then to find the word of least weight in the coset. We can use the parity check matrix to develop an easy way of doing this.

**Definition 2.51** Let $C$ be a linear code of length $n$ and dimension $k$. Let $\mathbf{H}$ be a parity check matrix for $C$. For any word $w \in K^n$, we define the *syndrome* of $w$ to be $w\mathbf{H}$, which is a word in $K^{n-k}$.

**Exercise 2.52** For the code $C$ defined in Example 2.50, a parity check matrix is given by

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

Calculate the syndrome of the received word $w = 1101$. Then calculate the syndrome of the word $u = 1000$ which is the word of least weight in the coset $C + w$.

Notice that the syndromes of $w$ and $u$ are the same. It is always the case that words from the same coset will have the same syndrome. Suppose we have a linear code $C$ with parity check matrix $\mathbf{H}$, and let $u$ be a word not in $C$. If $v \in C$ then the word $v + u$ lies in the same coset as $u$. The syndrome of $v + u$ is equal to the syndrome of $u$ since

$$(v + u)\mathbf{H} = v\mathbf{H} + u\mathbf{H} = \mathbf{0} + u\mathbf{H} = u\mathbf{H}.$$

Furthermore, in Exercise 2.52, notice that if $w = 1101$ is received, MLD concludes that $v = u + w = 1101 + 1000 = 0101$ was sent, so there was an error in the first digit. Notice also that for the error pattern $u$, the syndrome $uH$ is equal to the row of $H$ (the first) corresponding to the location of the most likely error. In this way the syndrome indicates the symptoms of the error.

The following theorem gives some basic and useful facts about the syndrome.

**Theorem 2.53** *Let $C$ be a linear code of length $n$, $\mathbf{H}$ be a parity check matrix for $C$ and $w, u \in K^n$. Then*

*(a) $w\mathbf{H} = \mathbf{0}$ iff $w$ is a codeword in $C$.*

*(b) $w\mathbf{H} = u\mathbf{H}$ iff $w$ and $u$ lie in the same coset of $C$.*

*(c) If $u$ is the error pattern in a received word $w$ then $u\mathbf{H}$ is the sum of the rows of $\mathbf{H}$ that correspond to the positions where errors occurred in transmission.*

Note that if no errors occur in transmission and $w$ is received then $w\mathbf{H} = \mathbf{0}$. But $w\mathbf{H} = \mathbf{0}$ does not necessarily imply that no errors occurred, since the codeword $w$ received is not necessarily the same as the codeword sent.

Since words in the same coset have the same syndrome and words in different cosets have different syndromes, we can identify a coset by the syndrome of any word in the coset. Thus for a $(n, k)$ code, the $2^{n-k}$ words of length $n - k$ each occurs as the syndrome of exactly one of the $2^{n-k}$ cosets.

To calculate the syndrome of a particular coset, we select a word $w$ in the coset and calculate the syndrome $w\mathbf{H}$. For MLD, we want a word of least weight in the coset to use as the error pattern. Any word of least weight in a coset is called a *coset leader*. If there is more than one candidate for coset leader in a coset, we choose one arbitrarily (for CMLD), or remember that we need to request retransmission if this syndrome arises (for IMLD).

**Example 2.54** Let $C$ be the code from Example 2.50, with parity check matrix $H$ given in Exercise 2.52. For each coset we can calculate the syndrome using the coset leader:

| Coset leader $u$ | Syndrome $uH$ |
|---|---|
| 0000 | 00 |
| 1000 | 11 |
| 0100 or 0001 | 01 |
| 0010 | 10 |

Note again that each word of length 2 occurs precisely once as a syndrome.

**Definition 2.55** The table in Example 2.54 which matches each syndrome with its coset leader(s) is called a *standard decoding array* or *SDA*. To construct an SDA, first list all of the cosets for the code and choose from each coset the word(s) of least weight as coset leader(s) $u$. Then find a parity check matrix $\mathbf{H}$ for the code and for each coset leader $u$ calculate its syndrome $u\mathbf{H}$.

**Exercise 2.56** Let $C = \{00000, 10100, 01011, 11111\}$. Construct an SDA for $C$.

First we calculate the cosets of $C$:

$$\begin{aligned} C &= \{00000,\ 10100,\ 01011,\ 11111\} \\ 10000 + C &= \{10000,\ 00100,\ 11011,\ 01111\} \\ 01000 + C &= \{01000,\ 11100,\ 00011,\ 10111\} \\ 00010 + C &= \{00010,\ 10110,\ 01001,\ 11101\} \\ 00001 + C &= \{00001,\ 10101,\ 01010,\ 11110\} \\ 11000 + C &= \{11000,\ 01100,\ 10011,\ 00111\} \\ 10010 + C &= \{10010,\ 00110,\ 11001,\ 01101\} \\ 10001 + C &= \{10001,\ 00101,\ 11010,\ 01110\} \end{aligned}$$

Having constructed an SDA, it is easy to find the codeword corresponding to the received word $w$ using MLD. We first calculate the syndrome $w\mathbf{H}$, then find the coset leader $u$ in the SDA with the same syndrome (so $u\mathbf{H} = w\mathbf{H}$). Then we conclude that $v = w + u$ was the most likely transmitted codeword, or request retransmission.

**Exercise 2.57** Let $C$ be the code given in Exercise 2.56. Use the method of syndrome decoding to find the codewords corresponding to each of the received words $w_1 = 10101$, $w_2 = 01110$ and $w_3 = 00011$.

We can check that for the previous example, the method of syndrome decoding agrees with decoding by chosing the closest codeword.

In Exercise 2.57, we decoded $w_1 = 10101$ to $v_1 = 10100$. By comparing the distance from $w_1$ to each of the codewords $d(00000, 10101) = 3$, $d(10100, 10101) = 1$, $d(01011, 10101) = 4$ and $d(11111, 10101) = 2$, we see that $v_1$ is the closest codeword to $w_1$.

In Exercise 2.57 we requested retransmission for the received word $w_2 = 01110$. This is the correct outcome since $d(11111, 01110) = 2$ and $d(01011, 01110) = 2$.

In practice, a code might have $2^{50}$ coset leaders and syndromes, which makes an SDA for an arbitrary linear code very large. Thus we still have not resolved decoding for an arbitrary linear code. However, for certain linear codes, finding the coset leaders is easy.

For example, consider the $(7, 4, 3)$ code with generating matrix $\mathbf{G}$ and parity check matrix $\mathbf{H}$.

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

This code has $2^4 = 16$ codewords, and so it has $2^{7-4} = 2^3 = 8$ cosets, each with 16 words. But note that there are exactly 8 words of length 7 with weight less than or equal to one. Also note that since this code has distance 3, it can correct up to one error. Thus each of the words of weight less than or equal to one must be a coset leader. Since we have 8 such words, no word of weight more than one is a coset leader, so it corrects only error patterns of weight one. We'll come back to a more detailed discussion of this code in a later section.

## 2.10 Reliability of IMLD for linear codes

Let $C$ be a linear code of length $n$ and dimension $k$. Recall that $\Theta_p(C, v)$ is the probability that if the codeword $v$ is sent over a binary symmetric channel with reliability $p$, then IMLD will correctly conclude that $v$ was sent. When we looked at reliability for a general code, we found that $\Theta_p(C, v)$ might depend on which codeword $v$ we use. For linear codes, the value of $\Theta_p(C, v)$ is the same for all codewords $v$ in $C$.

Recall that $L(v)$ is the set of received words for which the process of IMLD will conclude that $v$ was the codeword transmitted, and that

$$\Theta_p(C, v) = \sum_{w \in L(v)} \Phi_p(v, w)$$

where $\Phi_p(v, w) = p^{n-d}(1-p)^d$ and $v$ and $w$ differ in $d$ positions.

For each unique coset leader $u$ and for each codeword $v$ in $C$, the word $v + u$ is closer to $v$ than to any other codeword. Also if a word $w \neq v + u$ for some codeword $v$ and some unique coset leader $u$, then $w$ is at least as close to some other codeword as it is to $v$. Thus, for a linear code, the set $L(v)$

of words that are closer to $v$ than to any other codeword is

$$L(v) = \{w \mid w = v + u \text{ where } u \text{ is a unique coset leader}\}.$$

Note that if $v$ is the zero word, then $L(v) = L(\mathbf{0})$ is the set of unique coset leaders.

If $w = v + u$ then the probability that $w$ is received if $v$ is transmitted depends only on the weight of $u$

$$\Phi_p(v, w) = \Phi_p(v, v + u) = p^{n-\mathrm{wt}(u)}(1-p)^{\mathrm{wt}(u)},$$

so for a linear code $C$, the value of $\Theta_p(C, v)$ does not depend on the specific codeword $v$. For a linear code $C$, we denote $\Theta_p(C, v)$ by $\Theta_p(C)$ and

$$\Theta_p(C) = \Theta_p(C, v) = \sum_{w \in L(v)} \Phi_p(v, w) = \sum_{u \in L(\mathbf{0})} p^{n-\mathrm{wt}(u)}(1-p)^{\mathrm{wt}(u)}.$$

Thus, to calculate the reliability of IMLD for a linear code $C$, just calculate the probability of each unique coset leader occurring as an error pattern and then sum these probabilities to obtain $\Theta_p(C)$.

Notice that we have also shown that for a linear code, the set of error patterns that can be corrected using IMLD is equal to the set of unique coset leaders.

**Exercise 2.58** Let $C$ be the linear code from Exercise 2.56. Determine the reliability of IMLD for the code $C$ using a BSC with reliability $p = 0.97$.

## 2.11 Encoding and decoding summary

We now summarise the information transmission process for a linear code of length $n$ and dimension $k$. The possible messages are the $2^k$ words of length $k$, that is, all the vectors in the vector space $K^k$. The codewords are $2^k$ of the possible $2^n$ words of length $n$, and the code is a subspace of the vector space $K^n$. The received words are all the possible words of length $n$. Let $\mathbf{G}$ and $\mathbf{H}$ be a generating matrix and parity check matrix, respectively, for the linear code.

| | | | | |
|---|---|---|---|---|
| Message word $u$ of length $k$ | compute $u\mathbf{G}$ $\longrightarrow$ | Codeword $v = u\mathbf{G}$ of length $n$ | noisy channel $\longrightarrow$ | Received word $w$ of length $n$ |
| | | | | compute $w\mathbf{H}$ $\downarrow$ |
| Most likely message word is $u'$ | decode using $\mathbf{G}$ $\longleftarrow$ | Most likely codeword is $v' = w + e$ | compute $w + e$ $\longleftarrow$ | Most likely error pattern $e$ is the coset leader of the coset with syndrome $w\mathbf{H}$ |

Note that how you determine $u'$ depends on your generating matrix $\mathbf{G}$. If $\mathbf{G}$ is in RREF then you can find $u'$ by choosing the components of $v'$ that correspond to the leading columns of $\mathbf{G}$. Note also that depending on how many errors occurred during transmission over the channel, $u'$ may or may not be the same as $u$. If $\delta$ is the distance of $C$ and fewer than $\left\lfloor \frac{\delta - 1}{2} \right\rfloor$ errors occurred, then $u' = u$.

**Exercise 2.59** Let $C$ be the code with generating matrix $\mathbf{G}$ and parity check matrix $\mathbf{H}$ where

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

Write down the results of the encoding, transmission and decoding processes, starting with the message word $u = 0101$ and assuming that a single error in position 4 occurs during transmission.
Write down the results of the encoding, transmission and decoding processes, starting with the message word $u = 0101$ and assuming that two errors in positions 4 and 7 occur during transmission.

# 3 Linear codes II

In this section we look at some construction techniques for linear codes and investigate some important families of linear codes.

## 3.1 Some bounds for codes

We now look at the problem of determining how many words a linear code $C$ of length $n$ and distance $\delta$ can possibly have. This problem has not been solved in general, although it has been settled for certain values of $n$ and $\delta$. There are two well-known bounds for the maximum number of possible codewords for a given $n$ and $\delta$. Clearly we would like to choose codes with as many possible codewords for given values of $n$ and $\delta$.

The number of distinct words of length $n$ and weight $t$ is

$$\binom{n}{t} = \frac{n!}{t!(n-t)!}.$$

**Theorem 3.1** *If $0 \leq t \leq n$ and $v$ is a word of length $n$ then the number of distinct words of length $n$ and at distance at most $t$ from $v$ is precisely*

$$\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}.$$

**Proof:** Given a fixed word $v$ of length $n$, to find all words at distance exactly $i$ from $v$, we add to $v$ each word of weight $i$ and length $n$. There are $\binom{n}{i}$ such words. Letting $i$ range from 0 to $t$ gives the result. □

Let $C$ be a code of length $n$ and distance $\delta = 2t + 1$. Then there is no word $w$ at distance at most $t$ from any two distinct codewords. Thus the list of words at distance at most $t$ from any codeword $v_1$ must have no intersection with the list of words at distance at most $t$ from any other codeword $v_2$. Hence we have the following very important result.

**Theorem 3.2 The Hamming Bound**
*If $C$ is a code of length $n$ and distance $\delta = 2t + 1$ or $\delta = 2t + 2$ then*

$$\mid C \mid \left( \binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t} \right) \leq 2^n,$$

*or equivalently,*

$$\mid C \mid \leq \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}}.$$

The Hamming bound is an **upper** bound for the number of words in a code (linear or not) of length $n$ and distance $\delta = 2t + 1$. Note that such a code will correct all error patterns of weight less than or equal to $t$.

Recall that for a linear code, $\mid C \mid$ must be a power of 2.

**Exercise 3.3** Find an upper bound for the number of codewords in a linear code $C$ of length $n = 6$ and distance $\delta = 3$.

**Exercise 3.4** Investigate the Hamming bound for the $(7, 4, 3)$ code $C$ with generator matrix

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

(Note that this is the code that was discussed at the end of the Subsection 2.9.)

The next two results provide further inequalities involving the number of codewords in a linear code. The first gives a check on whether a linear code with given length, dimension and distance can possibly exist, and the second gives a **lower** bound for the maximum number of codewords in a linear code of given length and distance.

**Theorem 3.5** **The Gilbert-Varshamov Bound** *Let $n$, $k$ and $\delta$ be integers. If*

$$\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta-2} < 2^{n-k},$$

*then there exists a linear code of length $n$, dimension $k$ and distance at least $\delta$.*

**Proof:** Suppose that the integers $n$, $k$ and $\delta$ satisfy

$$\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta-2} < 2^{n-k}.$$

We will construct an $n \times (n-k)$ matrix $\mathbf{H}$ with linearly independent columns and in which no set of $\delta - 1$ or fewer rows is linearly dependent. This matrix can then be used as a parity check matrix to give a linear code which satisfies the theorem.

First note that the number of linear combinations of up to $x$ vectors chosen from a set of $y$ vectors is:

$$\binom{y}{0} + \binom{y}{1} + \binom{y}{2} + \ldots + \binom{y}{x}$$

since any such linear combination is obtained by choosing $i$ vectors out of the set of $y$ vectors and adding them together, where $0 \leq i \leq x$.

Start building the matrix $\mathbf{H}$ by choosing the first $n-k$ rows of $\mathbf{H}$ to be the rows of the matrix $\mathbf{I}_{n-k}$. This will guarantee that the columns of $\mathbf{H}$ are linearly independent, which we need for $\mathbf{H}$ to be a parity check matrix. Furthermore, we are guaranteed that so far, we have $n-k$ rows, no set of $\delta - 1$ of which are linearly dependent.

Now we continue to add rows, subject to the condition that each vector of length $n-k$ that we choose as a row is not a linear combination of $\delta - 2$ or fewer of the previous rows. At each stage we have an $i \times (n-k)$ matrix in which no set of $\delta - 1$ rows is linearly dependent. Then, for the $(i+1)$th row, we can choose any vector that is not a linear combination of up to $\delta - 2$ of the previous rows. That is, for the $(i+1)$th row, we can choose any one of

$$2^{n-k} - \left[\binom{i}{0} + \binom{i}{1} + \ldots + \binom{i}{\delta - 2}\right]$$

vectors. So, provided that

$$2^{n-k} > \left[\binom{i}{0} + \binom{i}{1} + \ldots + \binom{i}{\delta - 2}\right],$$

we can extend our $i \times (n-k)$ matrix to a $(i+1) \times (n-k)$ matrix in which no set of $\delta - 1$ rows is linearly depdendent.

Thus from an $(n-1) \times (n-k)$ matrix, we can build an $n \times (n-k)$ matrix, in which no set of $\delta - 1$ rows is linearly depdendent, provided that

$$2^{n-k} > \left[\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta - 2}\right].$$

This construction gives an $n \times (n-k)$ matrix $\mathbf{H}$ with linearly independent columns and in which no set of $\delta - 1$ rows is linearly dependent. Hence $\mathbf{H}$ can be used as a parity check matrix to give a linear $(n, k)$ code with distance at least $\delta$. □

**Corollary 3.6** *If $n \neq 1$ and $\delta \neq 1$ then there exists a linear code $C$ with length $n$ and distance at least $\delta$ with*

$$| C | \geq \frac{2^{n-1}}{\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta-2}}.$$

**Proof:** The Gilbert-Varshamov Bound guarantees the existence of a code $C$ with length $n$, dimension $k$ (so $|C| = 2^k$) and distance at least $\delta$ if

$$\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta - 2} < 2^{n-k}.$$

By multiplying both sides of this inequality by $2^k$ we see that a code $C$ with length $n$, dimension $k$ and distance at least $\delta$ exists if

$$2^k \left[ \binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta-2} \right] < 2^n.$$

Thus for any integers $n$, $k$ and $\delta$ ($n \neq 1$, $\delta \neq 1$), if

$$2^k < \frac{2^n}{\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta-2}}$$

then the Gilbert-Varshamov Bound guarantees the existence of a code with $2^k$ codewords. Let $k'$ be the *largest* value of $k$ for which this inequality is satisfied. Then

$$2^{k'} \geq \frac{2^{n-1}}{\binom{n-1}{0} + \binom{n-1}{1} + \ldots + \binom{n-1}{\delta-2}}$$

and there exists a code $C$ with $|C| = 2^{k'}$. □

Note that for a linear code $C$, we know that $|C|$ is a power of 2. So if $|C| \geq m$, where $m$ is not a power of 2, then we actually have $|C| \geq m'$, where $m'$ is the smallest power of 2 that is greater than $m$.

**Exercise 3.7** Does there exist a linear code of length $n = 9$, dimension $k = 2$ and distance $\delta = 5$?

**Exercise 3.8** Determine a lower and upper bound for the maximum number of codewords in a linear code with $n = 9$ and $\delta = 5$ (so $k$ is not restricted).

Note that the lower bound on $|C|$ given in Corollary 3.6 is a lower bound on the size of the **biggest** code of length $n$, dimension $k$ and distance $\delta$. It is not a bound that can be used for every linear code. For example, there exists a linear code of length $n = 9$ and distance $\delta = 5$, for which $|C| = 2$.

$$C = \{000000000, 111110000\}$$

What was shown in Exercise 3.8 was that there exists a better code with length 9 and distance 5. What the bounds tell us is that there must exist such a code with at least 4 codewords, and possibly that there exists such a code with 8 codewords.

## 3.2 Perfect codes

In the previous section we saw several bounds on the possible number of words in codes. Here we describe an important class of codes in which one of these bounds is attained.

**Definition 3.9** A code $C$ of length $n$ and odd distance $\delta = 2t + 1$ is called a *perfect code* if $C$ attains the Hamming bound given in Theorem 3.2. That is, $C$ is perfect iff

$$| C |= \frac{2^n}{\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}}.$$

There are not very many perfect codes, but ones that do exist are very useful. The main problem in finding perfect linear codes is that $|C|$ is a power of 2, so the expression

$$\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}$$

must also be a power of 2.

**Example 3.10** Let $t = 0$. Then $\binom{n}{0} = 1 = 2^0$, so $|C| = \frac{2^n}{\binom{n}{0}} = 2^n$. The only code with $2^n$ codewords and length $n$ is $K^n$, so $K^n$ is a perfect code. (Of course, this code provides no error detection or correction.)

**Example 3.11** Similarly, we can show that there is a perfect code of length $2t+1$ and distance $2t+1$ with 2 codewords: one codeword is the zero word, and the other is the word containing all 1s.

The codes from the two previous examples are not particularly interesting, and are called the *trivial* perfect codes.

In Exercise 3.4 we showed that there exists a perfect linear code with $n = 7$ and $\delta = 3$. It has 16 codewords. (Note that this code is the code discussed at the end of Subsection 2.9.)

**Exercise 3.12** The *Golay* code is a perfect code with $n = 23$ and $\delta = 7$. Show that such a code may exist, and find the number of codewords in it.

The possible lengths and distances for a perfect code were determined by Tietavairen and van Lint in 1963. They showed that:

**Theorem 3.13** *If $C$ is a non-trivial perfect code of length $n$ and distance $\delta = 2t+1$ then either $n = 23$ and $\delta = 7$, or $n = 2^r - 1$ for some $r \geq 2$ and $\delta = 3$.*

Note that Theorem 3.13 does not state that every code meeting those requirements will be perfect, just that every perfect code must satisfy those requirements.

Suppose $C$ is a linear code of length $n$ and distance $\delta = 2t + 1$. By Theorem 1.47 $C$ will correct all error patterns of weight less than or equal to $t$. Thus every word of length $n$ and weight less than or equal to $t$ is a coset leader. There are exactly

$$\binom{n}{0} + \binom{n}{1} + \ldots + \binom{n}{t}$$

such words, which is precisely the number of cosets if the code is perfect. Hence we have the following.

**Theorem 3.14** *If $C$ is a perfect code of length $n$ and distance $\delta = 2t + 1$ then $C$ will correct all error patterns of weight less than or equal to $t$, and no other error patterns.*

**Definition 3.15** A perfect code which corrects all error patterns of weight less than or equal to $t$ is called a *perfect $t$-error correcting code*. From Theorem 3.13, the only possible values for $t$ are $t = 1$ and $t = 3$.

## 3.3 Extended codes

Sometimes, increasing the length of a code by one digit or a few digits gives a new code with improved error detection or correction, which justifies the lower information rate.

**Definition 3.16** Let $C$ be a linear code of length $n$. The code $C^*$ of length $n + 1$, formed from $C$ by adding one extra digit to each $v \in C$ in order to make each $v^* \in C^*$ have even weight is called an *extended code of $C$*.

Note that since $C$ is a linear code, $C^*$ is also a linear code.

**Example 3.17** At the start of our discussions on coding theory, we formed extended codes of $K^2$ and $K^3$ by adding a single parity check digit to the end of each codeword so that each codeword had even weight.

**Theorem 3.18** Suppose that we form an extended code $C^*$ by adding a digit to the end of each codeword of a linear code $C$. If the original code $C$ has a $k \times n$ generating matrix $\mathbf{G}$ then the extended code $C^*$ has a $k \times (n + 1)$ generating matrix

$$\mathbf{G}^* = \begin{pmatrix} \mathbf{G} & b \end{pmatrix},$$

where the last column $b$ of $\mathbf{G}^*$ is appended so that each row of $\mathbf{G}^*$ has even weight.

A parity check matrix for $C^*$ can be constructed from $\mathbf{G}^*$ using Algorithm 2.13. However, the following theorem provides a quicker method to find a parity check matrix for $C^*$.

**Theorem 3.19** If $\mathbf{H}$ is a parity check matrix for $C$, then a parity check matrix for $C^*$ is given by

$$\mathbf{H}^* = \begin{pmatrix} \mathbf{H} & j \\ \mathbf{0} & 1 \end{pmatrix},$$

where $j$ is a column containing all 1s.
**Proof:** We have that $\mathbf{H}$ is an $n \times (n-k)$ matrix with rank $(n-k)$. Hence $\mathbf{H}^*$ is an $(n+1) \times (n+1-k)$ matrix with rank $(n-k+1)$. Moreover,

$$\mathbf{G}^*\mathbf{H}^* = \begin{pmatrix} \mathbf{G} & b \end{pmatrix} \begin{pmatrix} \mathbf{H} & j \\ \mathbf{0} & 1 \end{pmatrix} = \begin{pmatrix} \mathbf{GH} & \mathbf{G}j + b \end{pmatrix}.$$

Now $\mathbf{GH} = \mathbf{0}$ and $\mathbf{G}j$ sums the ones in each row of $\mathbf{G}$, so from the definition of $b$ it follows that $\mathbf{G}j + b = \mathbf{0}$. □

**Exercise 3.20** Let $C$ be the linear code with generating matrix $\mathbf{G}$ and parity check matrix $\mathbf{H}$, where

$$\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{H} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Find $\mathbf{G}^*$ and $\mathbf{H}^*$ for the extended code $C^*$.

If $C$ is an $(n, k, \delta)$ linear code, what can be said about the length, dimension, distance and rate of the extended code $C^*$?

The length and dimension of $C^*$ are $(n+1)$ and $k$, respectively. Thus, the rate of $C^*$ is $k/(n+1)$. Notice that $C^*$ is a slightly less efficient code than $C$ since

$$\frac{k}{n} > \frac{k}{n+1}.$$

Let $v \in C$ be a nonzero codeword of minimum weight in $C$. Thus $\delta = \mathrm{wt}(v)$. If $v^* \in C^*$ is the codeword corresponding to $v \in C$, then $\mathrm{wt}(v^*) = \mathrm{wt}(v)$ if $\mathrm{wt}(v)$ is even, and $\mathrm{wt}(v^*) = \mathrm{wt}(v) + 1$ if $\mathrm{wt}(v)$ is odd. Hence if $C$ has odd distance $\delta$, then the distance of $C^*$ is $\delta + 1$, whereas if $\delta$ is even, then the distance of $C^*$ is still $\delta$. Thus an extended code is only useful when the distance of the original code is odd, in which case the extended code corrects no more errors than the original code, but it detects one more error.

When forming an extended code, there is no particular reason to add the extra digit to the end of each original codeword. A code with similar properties can be constructed by inserting an extra digit in any particular position in each codeword (as long as it is the same position in each codeword).

**Exercise 3.21** Let $C$ be the linear code $C = \{000000, 111000, 000111, 111111\}$. Form an extended code $C^*$ by adding a parity check digit to the *start* of each codeword of $C$. A generating matrix and parity check matrix for $C$ are:

$$\mathbf{G}_C = \begin{pmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \qquad \mathbf{H}_C = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}.$$

Describe a generating matrix and parity check matrix for $C^*$ in terms of $\mathbf{G}_C$ and $\mathbf{H}_C$. What are the length, dimension, distance and information rate for each of the two codes?

## 3.4 The $(a \mid a+b)$ construction

Given two codes of length $n$, we can combine these to form a new code of length $2n$ with good error correcting properties.

**Example 3.22** Consider the following two codes of length $n = 4$:

$$\begin{array}{ll} \text{Code } A & \{0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111\} \\ \text{Code } B & \{0000, 1111\}. \end{array}$$

Define a new code $C$ with words of length 8. The first four bits of each new codeword are a codeword $a$ of $A$, and the last four bits of each codeword are obtained by adding to $a$ a codeword $b$ of $B$. Thus for each $a \in A$, there will be two new codewords in $C$ (as there are two codewords in $B$), each of the form $a \mid a+b$. The new code $C$ is:

| $a \mid a+0000$ | $a \mid a+1111$ |
|---|---|
| 00000000 | 00001111 |
| 00110011 | 00111100 |
| 01010101 | 01011010 |
| 01100110 | 01101001 |
| 10011001 | 10010110 |
| 10101010 | 10100101 |
| 11001100 | 11000011 |
| 11111111 | 11110000 |

**Theorem 3.23** *Let $A$ be an $(n, k_A, \delta_A)$ linear code, $B$ be an $(n, k_B, \delta_B)$ linear code and let $\delta$ be the smaller of $2\delta_A$ and $\delta_B$. Then the code $C$ whose codewords are all the binary words of the form $(a \mid a+b)$, where $a \in A$ and $b \in B$, is a $(2n, k_A + k_B, \delta)$ linear code.*

**Proof:** The length of code $C$ is clearly $2n$. Code $A$ contains $2^{k_A}$ codewords and code $B$ contains $2^{k_B}$ codewords, so the code $C$ contains $2^{k_A} \times 2^{k_B} = 2^{k_A+k_B}$ codewords.

To verify that $C$ has minimum distance $\delta = \min\{2\delta_A, \delta_B\}$, note firstly that $C$ contains all codewords of the form $(a \mid a+0)$ for $a \in A$ and all codewords of the form $(0 \mid 0+b)$ for $b \in B$. Hence, $\delta \leq \min\{2\delta_A, \delta_B\}$. However, if $C$ contains a nonzero codeword $c = (a \mid a+b)$, then either $b = 0$ and so $\mathrm{wt}(c) \geq 2\delta_A$, or $b \neq 0$ in which case

$$\mathrm{wt}(c) = \mathrm{wt}(a \mid a+b) = \mathrm{wt}(a) + \mathrm{wt}(a+b) = d(0,a) + d(a,b) \geq d(0,b)$$

by the triangle inequality and so $\mathrm{wt}(c) \geq \delta_B$. Thus in either case, $\mathrm{wt}(c) \geq \min\{2\delta_A, \delta_B\}$. □

**Example 3.24** In Example 3.22, the original codes $A$ and $B$ had $n = 4$, $k_A = 3$, $\delta_A = 2$, $k_B = 1$ and $\delta_B = 4$. The resulting code $C$ is an $(8, 4, 4)$ code. If the two codes $A$ and $B$ are interchanged in the construction, the resulting code is only an $(8, 4, 2)$ code.

**Theorem 3.25** *If $A$ is an $(n, k_A)$ code with generating matrix $\mathbf{G}_A$ and parity check matrix $\mathbf{H}_A$, and $B$ is an $(n, k_B)$ code with generating matrix $\mathbf{G}_B$ and parity check matrix $\mathbf{H}_B$, then a generatng matrix $\mathbf{G}_C$ and parity check matrix $\mathbf{H}_C$ for the code $C$ formed by the $(a \mid a+b)$ construction are*

$$\mathbf{G}_C = \begin{pmatrix} \mathbf{G}_A & \mathbf{G}_A \\ 0 & \mathbf{G}_B \end{pmatrix}, \qquad \mathbf{H}_C = \begin{pmatrix} \mathbf{H}_A & \mathbf{H}_B \\ 0 & \mathbf{H}_B \end{pmatrix}.$$

**Proof:** Clearly, $\mathbf{G}_C$ is a $(k_A + k_B) \times 2n$ matrix with rank $k_A + k_B$ which generates $C$. $\mathbf{H}_C$ is a $2n \times (2n - k_A - k_B)$ matrix with linearly independent columns, and we have

$$\mathbf{G}_C\mathbf{H}_C = \begin{pmatrix} \mathbf{G}_A\mathbf{H}_A + 0 & \mathbf{G}_A\mathbf{H}_B + \mathbf{G}_A\mathbf{H}_B \\ 0 + 0 & 0 + \mathbf{G}_B\mathbf{H}_B \end{pmatrix} = \mathbf{0}.$$

□

## 3.5 Hamming codes

**Definition 3.26** A linear code of length $n = 2^r - 1$, $r \geq 2$, having a parity check matrix $\mathbf{H}$ whose rows consist of all nonzero vectors of length $r$ is called a *Hamming code* of length $2^r - 1$.

**Example 3.27** One possibility for a parity check matrix $\mathbf{H}$ for a Hamming code of length 7 (so $r = 3$) is:

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

By Algorithm 2.13, a generating matrix $\mathbf{G}$ for a Hamming code of length 7 is therefore

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

Thus the code has dimension 4 and contains $2^4 = 16$ codewords. The distance of the code is $\delta = 3$. The information rate is $4/7$.

An $n \times r$ matrix $\mathbf{H}$ whose rows are all the non-zero binary words of length $r$ must contain each word of weight one as a row and hence must have $r$ linearly independent columns. Hence $\mathbf{H}$ is a parity check matrix for some linear code and by definition this code is a Hamming code. Furthermore, there are precisely $2^r - 1$ non-zero binary words of length $r$, so $n = 2^r - 1$. Thus a Hamming code has length $n = 2^r - 1$ and has dimension $n - r = 2^r - 1 - r$, so it contains $2^{2^r-1-r}$ codewords.

We can find the distance of any Hamming code. No row of $\mathbf{H}$ is the zero word, so no single row of $\mathbf{H}$ is linearly dependent. No two rows of $\mathbf{H}$ are equal, so no two rows of $\mathbf{H}$ are linearly dependent. Thus $C$ has distance at least 3. But we can clearly choose three rows of $\mathbf{H}$ (say $100\ldots0$, $0100\ldots0$, $1100\ldots0$) which form a linearly dependent set. Thus by Theorem 2.42, a Hamming code has distance $\delta = 3$.

We can investigate the Hamming bound for any Hamming code. For $n = 2^r - 1$ and $\delta = 3 = 2t + 1$ (so $t = 1$), we have

$$\frac{2^n}{\binom{n}{0} + \ldots \binom{n}{t}} = \frac{2^n}{\binom{n}{0} + \binom{n}{1}} = \frac{2^{2^r-1}}{1+n} = \frac{2^{2^r-1}}{1 + 2^r - 1} = 2^{2^r-1-r}.$$

Putting together the previous few observations, Theorem 3.14 tells us that *Hamming codes are perfect, single-error correcting codes.*

It is trivial to construct an SDA for a Hamming code. All error patterns of weight 1 are corrected, so every word of length $2^r - 1$ and weight one must be a coset leader. If $e$ is an error pattern then $e\mathbf{H}$ sums the rows of the parity check matrix $\mathbf{H}$ corresponding to positions in which errors occurred. Hence, since $\mathbf{H}$ has $2^r - 1$ rows, an SDA for a Hamming code must be given by:

| coset leader | syndrome |
|---|---|
| $000\ldots0$ | $000\ldots0$ |
| $\mathbf{I}_{2^r-1}$ | $\mathbf{H}$ |

**Example 3.28** For the Hamming code in Example 3.27, assume $w = 1101001$ is the received word. The syndrome is $w\mathbf{H} = 011$, which is the fourth row of $\mathbf{H}$. Thus the coset leader is the fourth row of $I_7$, $u = 0001000$. We conclude that the most likely codeword is $w + u = 1100001$.

**Exercise 3.29** Use the Hamming code in Example 3.27 to determine the most likely codeword corresponding to each of the received words $w_1 = 1101011$ and $w_2 = 1111111$.

A Hamming code of length $2^r - 1$ is a $(2^r - 1, 2^r - 1 - r, 3)$ linear code. The advantage of using a Hamming code of long length is that it has many codewords (the maximum number of codewords possible for a given length and distance 3). However, every Hamming code is only 1-error correcting, so they are not useful if there is a high probability of errors.

Since Hamming codes have distance 3 (odd), we can gain some extra error detection capability by forming the extended Hamming code. The extended Hamming code of length $2^r$ is a $(2^r, 2^r - 1 - r, 4)$ linear code. They are 1-error correcting and 3-error detecting codes. The $(8, 4, 4)$ extended Hamming code was constructed in Example 3.22.

**Definition 3.30** The dual of the Hamming code of length $2^r - 1$ is called the *simplex* code of length $2^r - 1$. It is a $(2^r - 1, r, 2^{r-1})$ linear code. A simplex code is an example of a constant-weight code, since each nonzero codeword in a simplex code has weight $2^{r-1}$.

**Exercise 3.31** Describe how you could generate the codewords in the simplex code of length 7.

## 3.6 Reed-Muller codes

These codes were introduced in the 1950s by I.S. Reed and D.E. Muller. One of these codes was used by the Mariner 9 space probe to send pictures of Mars back to earth. Rather than maintaining a fixed distance (and hence error correction capacity, as did the Hamming codes), Reed-Muller codes give an increased distance as the length of codewords increases.

**Mariner 9 mission to Mars**

Launch: May 30, 1971
Arrival: Nov 13, 1971
Mass: 998 kilograms
Science instruments: Wide and narrow angle cameras with digital tape recorder, infrared spectrometer and radiometer, ultraviolet spectrometer, radio occultation and celestial mechanics instruments.

Mariner 9 was launched successfully on May 30, 1971, and became the first artificial satellite of Mars when it arrived on November 13, 1971 and went into orbit, where it functioned in Martian orbit for nearly a year. Mariner 9 completed its final transmission on October 27, 1972.

Upon arrival, Mariner 9 observed that a great dust storm was obscuring the whole globe of the planet. Ground controllers sent commands to the spacecraft to wait until the storm had abated, the dust had settled, and the surface was clearly visible before compiling its global mosaic of high-quality images of the Martian surface. The storm persisted for a month, but after the dust cleared, Mariner 9 proceeded to reveal a very different planet than expected – one that boasted gigantic volcanoes and a grand canyon stretching 4,800 kilometers across its surface. More surprisingly, the relics of ancient riverbeds were carved in the landscape of this seemingly dry and dusty planet. Mariner 9 exceeded all primary photographic requirements by photo-mapping 100 percent of the planet's surface. The spacecraft also provided the first closeup pictures of the two small, irregular Martian moons: Phobos and Deimos.

Mariner 9 had a radio transmitter with a power of only 20 watts, and was transmitting over a distance of 84 million miles. Despite this, near-perfect pictures were obtained. Each picture transmitted by Mariner 9 is made up of over half a million tiny picture elements forming a rectangular array. Each picture element is a uniform shade of grey, the precise shade being specified by a 9-bit binary number. Thus each picture was represented using 5,250,000 bits, in grey-scale. Given the large distances, low power, poor reliability of electronics in the transmitter and receiver, and the general background noise of space, many errors occurred in transmission. Hence it was necessary to perform extensive error correction.

Each message was divided into packets of 6 bits, and then each string of 6 bits was encoded into a 32 bit word. Thus a picture represented by 5 and a quarter million bits was transmitted using over 5 times that number of bits. The encoding was done using the first-order Reed-Muller code of length $2^5$, $RM(1,5)$. This is a $(32,6,16)$ code, with 64 codewords, which can detect 15 errors and correct up to 7 errors. The rate of the $RM(1,5)$ code is only $6/32$. For each bit of information, it was necessary to transmit more than 4 bits of redundant information. However, since errors were very likely, and retransmission was not possible, this redundancy was necessary and the mission was a success.

Reed-Muller codes are linear codes of length $2^m$, for some integer $m \geq 0$. The dimension of the code (and hence the number of codewords) depends on the order $r$ of the Reed-Muller code. The $r$th order Reed-Muller code of length $2^m$ will be denoted $RM(r,m)$, where $0 \leq r \leq m$. We give a recursive definition of these codes.

**Definition 3.32** Define the $r$th order Reed-Muller code of length $2^m$, denoted $RM(r,m)$, as follows:

1. $RM(0,m)$ is the linear code of length $2^m$ consisting of the zero word and the all ones word.
2. $RM(m,m)$ is the linear code of length $2^m$ whose codewords are all the binary words of that length, so $RM(m,m) = K^{2^m}$.
3. $RM(r,m)$ for $0 < r < m$ is obtained using the $(a \mid a+b)$ construction where $A = RM(r, m-1)$ and $B = RM(r-1, m-1)$.

Note that there are other ways to construct the Reed-Muller codes and you may have seen another construction method for first-order Reed-Muller codes in MATH2302.

**Example 3.33** The Reed-Muller codes of lengths $2^0$, $2^1$ and $2^2$ are listed below.

| $m$ | Reed-Muller codes of length $2^m$ |
|---|---|
| 0 | $RM(0,0) = \{0, 1\}$ |
| 1 | $RM(0,1) = \{00, 11\}$<br>$RM(1,1) = \{00, 01, 10, 11\}$ |
| 2 | $RM(0,2) = \{0000, 1111\}$<br>$RM(1,2) = \{(a \mid a+b) \mid a \in \{00,01,10,11\}, b \in \{00,11\}\}$<br>$= \{0000, 0101, 1010, 1111, 0011, 0110, 1001, 1100\}$<br>$RM(2,2) = K^4$ |

Using Theorem 3.25, we can give a recursive definition for a generating matrix of $RM(r,m)$.

**Definition 3.34** Let $\mathbf{G}_{r,m}$ be a generating matrix for $RM(r,m)$.

1. For $r = 0$ we define $\mathbf{G}_{0,m} = \begin{pmatrix} 1 & 1 & \dots & 1 \end{pmatrix}$, that is the $1 \times 2^m$ matrix of ones.
2. For $r = m$, we define $\mathbf{G}_{m,m} = \begin{pmatrix} \mathbf{G}_{m-1,m} \\ 0 \dots 01 \end{pmatrix}$.
3. For $0 < r < m$, we define $\mathbf{G}_{r,m} = \begin{pmatrix} \mathbf{G}_{r,m-1} & \mathbf{G}_{r,m-1} \\ \mathbf{0} & \mathbf{G}_{r-1,m-1} \end{pmatrix}$.

**Example 3.35** The generating matrices for some small Reed-Muller codes are given below.

$$\mathbf{G}_{0,0} = \begin{pmatrix} 1 \end{pmatrix} \qquad \mathbf{G}_{0,1} = \begin{pmatrix} 1 & 1 \end{pmatrix} \qquad \mathbf{G}_{1,1} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$$

$$\mathbf{G}_{0,2} = \begin{pmatrix} 1 & 1 & 1 & 1 \end{pmatrix} \qquad \mathbf{G}_{1,2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{pmatrix} \qquad \mathbf{G}_{2,2} = \begin{pmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

**Exercise 3.36** Determine a generating matrix for each of $RM(1,3)$ and $RM(2,3)$.

Note that $RM(1,3)$ is the $(8,4,4)$ extended Hamming code that has appeared earlier in the course.

**Exercise 3.37** State the length, dimension and distance of each of the Reed-Muller codes $RM(r,m)$ for $m=3$.

| Code | | | | |
|---|---|---|---|---|
| Length | | | | |
| Dimension | | | | |
| Distance | | | | |

**Theorem 3.38** *The $r$th order Reed-Muller code $RM(r,m)$ defined above has the following properties:*

1. *$RM(r,m)$ has length $2^m$;*
2. *$RM(r,m)$ has dimension $k = \sum_{i=0}^{r} \binom{m}{i}$;*
3. *$RM(r,m)$ has distance $\delta = 2^{m-r}$;*
4. *$RM(r-1,m)$ is contained in $RM(r,m)$, where $r > 0$;*
5. *the code $RM(m-1-r,m)$ where $r < m$ is the dual of the code $RM(r,m)$.*

The proofs of these claims are by induction and may appear in your tutorial problems or on an assignment.

The rate of $RM(r,m)$ gets very small as $m$ gets large. However, the *distance* is very large for the length of the code, so these codes are useful in situations where errors are likely and when error detection/correction is very important.

## 3.7 Decoding of first-order Reed Muller codes

There is a fast decoding algorithm for first-order Reed-Muller codes, based on the recursive nature of these codes. The proof that this algorithm works is beyond the scope of this course. The algorithm uses some matrix trickery (Fast Hadamard Transform) to identify the nearest codeword, rather than working with syndromes and coset leaders.

**Definition 3.39** The *Kronecker product* of two matrices $\mathbf{A}$ and $\mathbf{B}$ is defined by

$$\mathbf{A} \times \mathbf{B} = (a_{ij}\,\mathbf{B});$$

that is, entry $a_{ij}$ of matrix $\mathbf{A}$ is replaced by the matrix $a_{ij}\mathbf{B}$.

**Example 3.40** Let $\mathbf{L} = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$ and let $\mathbf{I}_n$ be the $n \times n$ identity matrix. Then

$$\mathbf{I}_2 \times \mathbf{L} = \begin{pmatrix} 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \quad \text{and} \quad \mathbf{L} \times \mathbf{I}_2 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{pmatrix}.$$

**Definition 3.41** We define a series of matrices based on the matrix $\mathbf{L}$ from Example 3.40. Let $m$ be a positive integer. For $i = 1, 2, \ldots, m$, we define the matrix $\mathbf{L}_m^i$ as

$$\mathbf{L}_m^i = \mathbf{I}_{2^{m-i}} \times \mathbf{L} \times \mathbf{I}_{2^{i-1}}.$$

**Example 3.42** Let $m = 2$. Then we have constructed the two matrices $\mathbf{L}_2^1$ and $\mathbf{L}_2^2$ in Example 3.40 since

$$\mathbf{L}_2^1 = \mathbf{I}_2 \times \mathbf{L} \times \mathbf{I_1} = \mathbf{I}_2 \times \mathbf{L}$$

$$\text{and} \quad \mathbf{L}_2^2 = \mathbf{I}_1 \times \mathbf{L} \times \mathbf{I_2} = \mathbf{L} \times \mathbf{I}_2$$

**Exercise 3.43** Determine the matrix $\mathbf{L}_3^1$.

The matrices $\mathbf{L}_3^2$ and $\mathbf{L}_3^3$ are given here for your reference.

$$\mathbf{L}_3^2 = \mathbf{I}_2 \times \mathbf{L} \times \mathbf{I}_2 = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & -1 \end{pmatrix}$$

$$\mathbf{L}_3^3 = \mathbf{I}_1 \times \mathbf{L} \times \mathbf{I}_4 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \end{pmatrix}$$

Before giving the decoding algorithm, recall that to write a natural number $x$ in binary representation with low order digits first, you write

$$x = \alpha_0 2^0 + \alpha_1 2^1 + \alpha_2 2^2 + \cdots + \alpha_i 2^i$$

where $i$ is the largest value such that $2^i \leq x$ and each $\alpha_j \in \{0, 1\}$, and then the binary representation of $x$ is $\alpha_0\alpha_1\alpha_2 \ldots \alpha_i$.

The following decoding algorithm incorporates both error detection/correction and the recovery of the intended message word, so it takes a received word of length $2^m$ and returns a message word of length $m + 1$.

**Algorithm 3.44 Decoding** *$RM(1, m)$ Suppose $w$ is the received word and $\mathbf{G}_{1,m}$ is the generating matrix for the $RM(1, m)$ code.*

1. *Replace each 0 in $w$ by $-1$ to form the word $\overline{w}$.*
2. *Calculate $w_1 = \overline{w}\mathbf{L}_m^1$ and then for each $i = 2, 3, \ldots, m$ calculate $w_i = w_{i-1}\mathbf{L}_m^i$.*
3. *Determine the position $j$ of the largest component (in absolute value) of $w_m$, where position is measured from left to right and counted from 0 to $2^m - 1$.*
4. *Let $v(j)$ be the binary representation of $j$ (low order digits first).*

*If the $j$th component of $w_m$ is positive, then the most likely intended message word is $(1, v(j))$. If the $j$th component is negative, then the most likely intended message word is $(0, v(j))$.*

**Example 3.45** A message has been encoded using the code $RM(1, 3)$. Determine the most likely intended message word if we receive the word $w = 10101011$.

We apply Algorihtm 3.44, and write our vectors using commas for clarity.

Convert $w$ to $\overline{w} = (1, -1, 1, -1, 1, -1, 1, 1)$. Compute

$$\begin{aligned} w_1 &= \overline{w}\mathbf{L}_3^1 = (0, 2, 0, 2, 0, 2, 2, 0) \\ w_2 &= w_1\mathbf{L}_3^2 = (0, 4, 0, 0, 2, 2, -2, 2) \\ w_3 &= w_2\mathbf{L}_3^3 = (2, 6, -2, 2, -2, 2, 2, -2) \end{aligned}$$

The largest component of $w_3$ is 6, occurring in position 1. Since $v(1) = 100$ and $6 > 0$, the presumed message word is 1100 (with corresponding codeword 10101010).

**Exercise 3.46** A message has been encoded using the code $RM(1,3)$. Apply Algorithm 3.44 to determine the most likely transmitted codeword if we receive the word $w = 10001111$.

## 3.8 The Extended Golay Code

The Voyager space mission (Voyager 1 was launched on 1 September 1977 and Voyager 2 was launched on 20 August 1977) successfully explored the planets Jupiter, Saturn, Uranus and Neptune and the two spacecraft continue their exploration as they move towards interstellar space. The Galileo spacecraft was launched on 18 October 1989, entered into orbit around Jupiter in 1995, and conducted a thorough exploration of Jupiter and its moons before making a mission-ending plunge into the planet in 2003. To send data back to Earth, each of these spacecraft used a pair of codes, one of which was the Extended Golay Code (a three error correcting code).

If we take the first-order Reed-Muller code $RM(1,3)$ and rearrange the bits in each codeword into the order $x_4x_6x_3x_2x_1x_5x_7x_8$, then we obtain the codewords of an equivalent code $A$ with generating matrix

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

If we then reverse the order of the first seven bits of each codeword of $A$, we obtain a code $B$ which is still equivalent to $RM(1,3)$. A generating matrix for $B$ is

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

We can then use codes $A$ and $B$ to construct two new important codes (first published in 1949). The construction is similar to the $(a \mid a + b)$ construction given earlier.

**Definition 3.47** The *extended Golay code* is the code whose codewords are all of the binary words which can be written in the form

$$(a_1 + b \mid a_2 + b \mid a_1 + a_2 + b)\,,$$

where $a_1, a_2 \in A$ and $b \in B$. Equivalently, each codeword in the extended Golay code can be written as the sum of three binary words

$$(a_1 \mid 0 \mid a_1) + (0 \mid a_2 \mid a_2) + (b \mid b \mid b)\,.$$

If $\mathbf{G}_A$ is a generating matrix for code $A$ and $\mathbf{G}_B$ is a generating matrix for code $B$ then a generating matrix for the extended Golay code is the $12 \times 24$ array

$$\mathbf{G} = \begin{pmatrix} \mathbf{G}_A & 0 & \mathbf{G}_A \\ 0 & \mathbf{G}_A & \mathbf{G}_A \\ \mathbf{G}_B & \mathbf{G}_B & \mathbf{G}_B \end{pmatrix}.$$

Of more use is the following version of the generating matrix, which is in standard form.

**Definition 3.48** Let $\mathbf{B}$ be the $12 \times 12$ matrix:

$$\mathbf{B} = \begin{pmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \end{pmatrix}.$$

and let $\mathbf{G}$ be the $12 \times 24$ matrix $\mathbf{G} = (\mathbf{I}_{12} \;\; \mathbf{B})$. The linear code $C_{24}$ having $\mathbf{G}$ as generating matrix is the *extended Golay code*, which we will denote $C_{24}$.

Note that if $\mathbf{B}_1$ is the $11 \times 11$ matrix obtained from $\mathbf{B}$ by deleting the last row and column, then $\mathbf{B}_1$ has a cyclic left-shift structure. Note also that the positions of the ones in the first 11 places of the first row correspond to the quadratic residues mod 11 (including 0). Also, note that $\mathbf{B}^T = \mathbf{B}$.

**Theorem 3.49 (Properties of $C_{24}$)** We note the following important facts about $C_{24}$.

1. $C_{24}$ has length 24 and dimension 12, with $|C_{24}| = 2^{12} = 4096$.
2. $C_{24}$ has parity check matrix $\mathbf{H} = \begin{pmatrix} \mathbf{B} \\ \mathbf{I}_{12} \end{pmatrix}$.
3. Another parity check matrix for $C_{24}$ is $\mathbf{H}' = \begin{pmatrix} \mathbf{I}_{12} \\ \mathbf{B} \end{pmatrix}$.
4. Another generating matrix for $C_{24}$ is $\mathbf{G}' = \begin{pmatrix} \mathbf{B} & \mathbf{I}_{12} \end{pmatrix}$.
5. $C_{24}$ is self-dual, so $C_{24}^{\perp} = C_{24}$.
6. The distance of $C_{24}$ is 8.
7. $C_{24}$ is a 3-error correcting code.
8. The weight distribution table for the codewords of $C_{24}$ is:

| wt | 0 | 4 | 8 | 12 | 16 | 20 | 24 |
|---|---|---|---|---|---|---|---|
| # words | 1 | 0 | 759 | 2576 | 759 | 0 | 1 |

**Proof of (2):** $\mathbf{H}$ is a $24 \times 12$ matrix with linearly independent columns, and

$$\mathbf{GH} = \begin{pmatrix} \mathbf{I}_{12} & \mathbf{B} \end{pmatrix} \begin{pmatrix} \mathbf{B} \\ \mathbf{I}_{12} \end{pmatrix} = \mathbf{B} + \mathbf{B} = \mathbf{0}.$$

**Proof of (3):** Note that $\mathbf{BB} = \mathbf{I}_{12}$. Thus $\mathbf{GH}' = \begin{pmatrix} \mathbf{I}_{12} & \mathbf{B} \end{pmatrix} \begin{pmatrix} \mathbf{I}_{12} \\ \mathbf{B} \end{pmatrix} = \mathbf{I}_{12}^2 + \mathbf{B}^2 = \mathbf{I}_{12} + \mathbf{I}_{12} = \mathbf{0}$.

**Proof of (4):** The new generating matrix has the correct number of rows and columns, and rows are linearly independent. Also, we have $\mathbf{G}'\mathbf{H}' = \mathbf{0}$.

**Proof of (5):** Note that $\mathbf{I}^T = \mathbf{I}$, $\mathbf{B}^T = \mathbf{B}$ and that every row of $\mathbf{G}$ has even weight. Hence $\mathbf{GG}^T = \mathbf{0}$.

**Proof of (6):** There are three steps:

I. *Proof that if $w \in C$, then $4 \mid \mathrm{wt}(w)$:*
Let $r_i$ denote row $i$ of $\mathbf{G}$. Any codeword $w$ can be expressed as the sum of up to 12 rows of $\mathbf{G}$, so $w = r_{i_1} + \ldots + r_{i_m}$, where $m \leq 12$.
Perform induction on $m$:
If $m = 1$ then $\mathrm{wt}(w) = \mathrm{wt}(r_{i_1}) = 8$ or 12, as can be seen from $\mathbf{G}$.
If $m \geq 1$ then assume the hypothesis is true for $m$ and consider $m + 1$.
We know $w = (r_{i_1} + \ldots + r_{i_m}) + r_{i_{m+1}} = v + r_{i_{m+1}}$, where $v$ is a codeword which is a sum of $m$ rows of $\mathbf{G}$ and by our hypothesis $4 \mid \mathrm{wt}(v)$. Since the rows of $\mathbf{G}$ are mutually orthogonal, $v$ and $r_{i_{m+1}}$ are orthogonal. Hence $v$ and $r_{i_{m+1}}$ have $2x$ ones in common. Therefore

$$\mathrm{wt}(w) = \mathrm{wt}(v) + \mathrm{wt}(r_{i_{m+1}}) - 2(2x).$$

Hence $4 \mid \mathrm{wt}(w)$.
So, by induction, statement I is true.

II. *Proof that the distance of $C_{24} = 4$ or 8:*
Since there are rows of $\mathbf{G}$ with weight 8, $d(C_{24}) \leq 8$. Clearly, the distance of $C_{24} \geq 1$, so the result follows from I.

III. *Proof that the distance of $C_{24} = 8$:*
Suppose we have a codeword $w$ of weight 4. Then, using our two generating matrices,

$$w = u_1 (\mathbf{I}_{12} \ \ \mathbf{B}) = u_2 (\mathbf{B} \ \ \mathbf{I}_{12})$$

for some $u_1$ and $u_2$. Write $w = [w_1, w_2]$ where $w_1$ and $w_2$ have length 12. Hence one of these has weight $\leq 2$. But $w_1 = u_1$ and $w_2 = u_2$. So one of $u_1$ or $u_2$ has weight $\leq 2$. Thus $w$ is the sum of either 1 or 2 rows of a generating matrix. However, the sum of any 2 rows of $\mathbf{B}$ has weight at least 6. Thus, as $\mathbf{B}$ forms part of the generating matrix concerned, $\mathrm{wt}(w) \geq 6$. Thus there are no words of weight 4, so the distance of $C_{24} = 8$. □

**Proof of (7):** Follows from (6).

**Explanation of (8):** The sum of all rows of $\mathbf{G}$ is $11\ldots1$, so $11\ldots1 \in C$. Hence the number of words of weight $k$ equals the number of words of weight $24 - k$. Hence it is easy to verify some of the entries in the weight table for $C_{24}$. Clearly, there is one word of weight zero and one word of weight 24. There are no words of weight 4 (as $\delta = 8$), so there are no words of weight 20.

## 3.9 Decoding the Extended Golay Code

We now define an algorithm for IMLD for $C_{24}$. Throughout this section, we use the following notation:

| | |
|---|---|
| $w$ | Received word |
| $v$ | Closest codeword to $w$ |
| $u$ | Error pattern ($u = v + w$) |

and we assume that message words of length 12 have been encoded using the generating matrix $\mathbf{G} = (\mathbf{I}_{12} \ \ \mathbf{B})$.

Our aim is to determine the coset leader $u$ of the coset containing $w$ without having to refer to the SDA of $C_{24}$. Since $C_{24}$ has distance 8, every error pattern of weight less than or equal to 3 must be a coset leader, so suppose we have an error pattern $u$ where $\mathrm{wt}(u) \leq 3$. Write $u = [u_1, u_2]$ where $u_1$ and $u_2$ each have length 12.

As $\mathrm{wt}(u) \leq 3$, we must have either $\mathrm{wt}(u_1) \leq 1$ or $\mathrm{wt}(u_2) \leq 1$. Let $\mathbf{H}_1$ be the parity check matrix

$$\mathbf{H}_1 = \begin{pmatrix} \mathbf{I}_{12} \\ \mathbf{B} \end{pmatrix}.$$

Then we have the syndrome $s_1 = w\mathbf{H}_1 = u\mathbf{H}_1 = u_1\mathbf{I}_{12} + u_2\mathbf{B} = u_1 + u_2\mathbf{B}$.
If $\mathrm{wt}(u_2) = 0$ then $s_1 = u_1$ so $\mathrm{wt}(s_1) \leq 3$ (as $\mathrm{wt}(u_1) \leq 3$).
If $\mathrm{wt}(u_2) = 1$ then $s_1 = u_1 +$ (1 row of $\mathbf{B}$), so $s_1$ is a row of $\mathbf{B}$ with at most 2 digits changed (as $\mathrm{wt}(u_1) \leq 2$).

Similarly, let $\mathbf{H}_2$ be the parity check matrix

$$\mathbf{H}_2 = \begin{pmatrix} \mathbf{B} \\ \mathbf{I}_{12} \end{pmatrix}.$$

In this case we have the syndrome $s_2 = w\mathbf{H}_2 = u\mathbf{H}_2 = u_1\mathbf{B} + u_2\mathbf{I}_{12} = u_1\mathbf{B} + u_2$.
If $\text{wt}(u_1) = 0$ then $s_2 = u_2$ so $\text{wt}(s_2) \leq 3$ (as $\text{wt}(u_2) \leq 3$).
If $\text{wt}(u_1) = 1$ then $s_2 = u_2+$ (1 row of $\mathbf{B}$), so $s_2$ is a row of $\mathbf{B}$ with at most 2 digits changed (as $\text{wt}(u_2) \leq 2$).

In any case, if $u$ has weight at most 3 then it is easily identified, since at most 3 rows of one of the two parity check matrices can be found to add to the corresponding syndrome. There are several possibilities corresponding to the possible weights of $u_1$ and $u_2$.

- If $\text{wt}(s_1) \leq 3$ then we have $u = [u_1, 0]$. (Here $\text{wt}(u_2) = 0$.)
- If $\text{wt}(s_2) \leq 3$ then we have $u = [0, u_2]$. (Here $\text{wt}(u_1) = 0$.)
- If $\text{wt}(s_1) \geq 3$ and $\text{wt}(s_2) \geq 3$, then we look for the row of $\mathbf{B}$ which is closest to $s_1$ or $s_2$ and use this to calculate the error pattern $u$.

These possibilities can be used to find an algorithm for decoding. However, in the decoding process we want to use only one parity check matrix. We will use $\mathbf{H} = \mathbf{H}_1$, but noting that $\mathbf{B}^2 = \mathbf{I}_{12}$, we have

$$\begin{aligned} s_2 &= u_1\mathbf{B} + u_2 \\ &= u_1\mathbf{B} + u_2\mathbf{I}_{12} \\ &= u_1\mathbf{B} + u_2\mathbf{B}^2 \\ &= (u_1 + u_2\mathbf{B})\mathbf{B} \\ &= s_1\mathbf{B}. \end{aligned}$$

In the following algorithm, $e_i$ is the word of length 12 with a one in the $i$th position and zeros elsewhere.

**Algorithm 3.50 IMLD for the Extended Golay Code, $C_{24}$**

1. Compute the syndrome $s = w\mathbf{H} = w\begin{pmatrix} \mathbf{I}_{12} \\ \mathbf{B} \end{pmatrix}$.
2. If $\text{wt}(s) \leq 3$ then $u = [s, 0]$.
3. If $\text{wt}(s + b_i) \leq 2$ for some row $b_i$ of $\mathbf{B}$ then $u = [s + b_i, e_i]$.
4. Compute the second syndrome $s\mathbf{B}$.
5. If $\text{wt}(s\mathbf{B}) \leq 3$ then $u = [0, s\mathbf{B}]$.
6. If $\text{wt}(s\mathbf{B} + b_i) \leq 2$ for some row $b_i$ of $\mathbf{B}$ then $u = [e_i, s\mathbf{B} + b_i]$.
7. If $u$ is not yet determined, then more than 3 errors have occurred so request retransmission.

Of course, once $u$ has been determined, $w$ is decoded to $w + u$.
Algorithm 3.50 requires at most 26 weight calculations in the decoding procedure. Of course, as soon as $u$ has been determined, no further steps in the algorithm need to be applied.

**Example 3.51** Decode the received word $w = 101\ 111\ 101\ 111\ 010\ 010\ 010\ 010$.
The syndrome is

$$s = w\mathbf{H} = 101\ 111\ 101\ 111 + 001\ 111\ 101\ 110 = 100\ 000\ 000\ 001,$$

which has weight 2. Since $\mathrm{wt}(s) \leq 3$, we find that $u = [s, 0] = [100\ 000\ 000\ 001, 0]$, and conclude that $v = w + u = 001\ 111\ 101\ 110\ 010\ 010\ 010\ 010$ was the codeword sent.

Because $\mathbf{G} = \begin{pmatrix} \mathbf{I}_{12} & \mathbf{B} \end{pmatrix}$ is in standard form and any word in $K^{12}$ can be encoded as a message ($C_{24}$ has dimension 12), the actual message sent occurs in the first 12 digits of the corresponding codeword $v$. Hence in the previous example, 001 111 101 110 was the message encoded and sent.

**Example 3.52** Decode the received word $w = 001\ 001\ 001\ 101\ 101\ 000\ 101\ 000$.
The syndrome is

$$s = w\mathbf{H} = 001\ 001\ 001\ 101 + 111\ 000\ 000\ 100 = 110\ 001\ 001\ 001,$$

which has weight 5. Proceeding to Step 3 of Algorithm 3.50, we compute

$$\begin{aligned} s + b_1 &= 000\ 110\ 001\ 100 \\ s + b_2 &= 011\ 111\ 000\ 010 \\ s + b_3 &= 101\ 101\ 011\ 110 \\ s + b_4 &= 001\ 001\ 100\ 100 \\ s + b_5 &= 000\ 000\ 010\ 010 \end{aligned}$$

Then, since $\mathrm{wt}(s + b_5) \leq 2$, we have that

$$u = [s + b_5, e_5] = [000\ 000\ 010\ 010,\ 000\ 010\ 000\ 000],$$

and conclude that $v = w + u = 001\ 001\ 011\ 111\ 101\ 010\ 101\ 000$ was the codeword sent.

**Exercise 3.53** Decode the received word $w = 000\ 111\ 000\ 111\ 011\ 011\ 010\ 000$.

Note that since there are $2^{n-k} = 2^{12} = 4096$ cosets and only $\binom{24}{0} + \binom{24}{1} + \binom{24}{2} + \binom{24}{3} = 2325$ cosets that have unique coset leaders of weight 3 or less, the code $C_{24}$ may correct some error patterns of higher weight. We will discuss this further in a moment when we look at the report "Decoding the Golay Code" by E.R.Berlekamp (obtained from the NASA website).

## 3.10 The Golay Code, $C_{23}$

**Definition 3.54** Another interesting 3-error-correcting code can be obtained by deleting the last digit from each word in $C_{24}$. This gives the *Golay Code*, $C_{23}$.

It is easily seen that $C_{24}$ is the extended code of $C_{23}$.

**Theorem 3.55** $C_{23}$ has the following properties:

1. A generating matrix is $(\mathbf{I}_{12}\ \hat{\mathbf{B}})$, where $\mathbf{B} = (\hat{\mathbf{B}}\ k)$ and $k = (11\ldots10)^T$. (That is, $\hat{\mathbf{B}}$ is obtained from the matrix $\mathbf{B}$ defined in the previous section by deleting the last column of $\mathbf{B}$.)
2. $C_{23}$ has length 23, dimension 12 (so $2^{12} = 4096$ codewords) and distance 7.
3. $C_{23}$ is a perfect 3-error correcting code.

**Proof of (3):** There are $2^{12}$ codewords and the entire vector space has $2^{23}$ words. Since $2^{23}/2^{12} = 2^{11}$, we must prove that there are exactly $2^{11}$ words of distance $\leq 3$ from a given codeword. But the number of such words is:

$$\begin{aligned} 1 + \tbinom{23}{1} + \tbinom{23}{2} + \tbinom{23}{3} &= 1 + 23 + 23 \cdot 11 + 23 \cdot 11 \cdot 7 \\ &= 2048 = 2^{11}. \end{aligned}$$

Thus $C_{23}$ is perfect. □

Thus $C_{23}$ will correct all error patterns of weight 3 or less, and no other error patterns (Theorem 3.14).

**Algorithm 3.56 IMLD for the Golay Code, $C_{23}$**

1. *Form $w^* = w0$ or $w^* = w1$ so that $w^*$ has odd weight.*
2. *Decode $w^*$ to a codeword $c^* \in C_{24}$ using Algorithm 3.50.*
3. *Remove the last digit from $c^*$ to obtain a codeword $c \in C_{23}$.*

Both the Golay code and the extended Golay code are 3-error correcting. To verify that Algorithm 3.56 is a valid algorithm, we need to check that if $v \in C_{23}$ is transmitted and $w$ is received, where $d(v, w) \leq 3$, then $d(v^*, w^*) \leq 3$ where $v^* \in C_{24}$ is the extension of $v$ and $w^*$ is as defined in Step 1 of Algorithm 3.56.
Suppose $v \in C_{23}$ is the codeword corresponding to the received word $w$, so $d(v, w) \leq 3$. Form $w^*$ by adding a digit to the end of $w$ such that $\mathrm{wt}(w^*)$ is odd. Let $v^* \in C_{24}$ be the extension of $v \in C_{23}$. Since $v^* \in C_{24}$, $\mathrm{wt}(v^*)$ is even. The difference between a word of odd weight and a word of even weight must be odd, so

$$d(v^*, w^*) \in \{1, 3, 5, 7, \ldots\}.$$

Also $d(v^*, w^*)$ is either $d(v, w)$ or $d(v, w) + 1$. Since $d(v, w) \leq 3$, we conclude that $d(v^*, w^*) \in \{1, 3\}$ so Algorithm 3.50 can be applied to $w^*$ to find $v^*$ and hence Algorithm 3.56 can decode $w$ to $v$.

Note that:

- As $C_{23}$ is perfect, there is never a need for retransmission.
- If $w$ is a codeword of $C_{23}$, then the syndrome of $w^*$ is the last row of $\mathbf{H}$. This case should be checked before applying Algorithm 3.50.

The extended Golay code is a $(24, 12, 8)$ code, hence providing 3 error correction (7 detection) and with rate 12/24. The Golay code is a $(23, 12, 7)$ code with 3 error correction (6 detection) and rate 12/23.

We have now met all of the non-trivial perfect binary codes: the Hamming codes and the Golay code. There are no others.

# 4 Cyclic Codes

## 4.1 Introduction to burst errors

Until now, we have assumed that errors occuring in transmission over a binary symmetric channel are randomly distributed. However, in some channels, it may be likely that errors occur very close to each other. For example, a compact disc may become scratched, or solar radiation may cause a group of errors which occur close together. In this section we consider codes that are designed to cope with this type of error pattern.

**Definition 4.1** Let $v$ be a word of length $n$. The *cyclic shift* of $v$, denoted $\gamma(v)$, is the word obtained by moving the last digit of $v$ to the beginning.

**Definition 4.2** A *burst* is a word that begins and ends with a one or is the empty word. A *burst error pattern* is a word of the form $0 \dots 0b0 \dots 0$ where $b$ is a burst. (Note that the 0s do not necessarily occur).
Let $B_n(x)$ denote the set of all burst error patterns belonging to $K^n$ with burst of length at most $x$, and let $C(B_n(x))$ denote the set of all cyclic shifts of words in $B_n(x)$. The words in $C(B_n(x))$ are called *cyclic* burst error patterns.

Note that the burst $0 \dots 0$ is defined to have length 0, so $0 \dots 0 \in B_n(x)$ always.

**Exercise 4.3** Determine $B_4(3)$ and $C(B_4(3))$.

**Example 4.4** Consider the words $v = 1101010$, $w_1 = 0100010$ and $w_2 = 0011010$. If we transmitted $v$ and $w_1$ was received, then the error pattern $e_1 = 1001000$ occurred, which has weight 2 and burst error length 4. If we transmitted $v$ and $w_2$ was received, then the error pattern $e_2 = 1110000$ occurred, which has weight 3 and burst error length 3.
Thus, if we transmit $v$ and if errors occur randomly on the channel, then $w_1$ is more likely than $w_2$ to be the received word. However, if we transmit $v$ and if errors occur in bursts on the channel, then $w_2$ is more likely than $w_1$ to be the received word.

When applying MLD to a linear code with the assumption of random errors, we constructed an SDA with coset leaders that were the words of least *weight* in the cosets, and said that such a code is $t$-error correcting precisely when all the words of weight at most $t$ are in different cosets of the code but not all of the words of weight at most $t + 1$ are in different cosets. If errors are likely to occur in bursts, then in applying MLD we construct an SDA with coset leaders that are the error patterns with burst of least *length* in each coset.

**Definition 4.5** A code $C$ is an *$x$ burst error correcting code* if all the words in $B_n(x)$ are in different cosets of $C$, but not all those in $B_n(x+1)$. An *$x$ cyclic burst error correcting code* is defined similarly: all words in $C(B_n(x))$ must be in different cosets, but not all of the words in $C(B_n(x+1))$ are in different cosets.

In general, if $C$ is $t$-error correcting and $x$ burst error correcting then $t \leq x$.

**Exercise 4.6** Show that it is impossible to have a 3-error correcting linear (15,9) code, but that it may be possible to have a 3 cyclic burst error correcting linear (15,9) code.

**Lemma 4.7** *If $C$ is a $t$ burst error correcting code of length $n$ and dimension $k$, then $t \leq n-k$.*

**Proof:** $C$ must correct all errors whose non-zero entries occur in the first $t$ places. There are $2^t$ of these, all of which must be in different cosets. But there are $2^{n-k}$ cosets in all.
Hence $2^t \leq 2^{n-k}$ and so $t \leq n-k$. □

Cyclic linear codes are a large family of codes that have a straightforward decoding algorithm for correcting cyclic burst error patterns. In order to define cyclic linear codes, we need to first remind ourselves of the connection between polynomials over $K$ and words of $K^n$.

## 4.2 Polynomials over $K$ and words of $K^n$

Recall the following facts from the cryptography part of the course.

**Definition 4.8** A *polynomial of degree $n$ over $K$* is a polynomial

$$\alpha_0 + \alpha_1 x + \ldots + \alpha_{n-1}x^{n-1} + \alpha_n x^n,$$

where the coefficients $\alpha_i \in K$ and $\alpha_n \neq 0$. The set of all polynomials over $K$ is denoted $K[x]$.

Polynomials over $K$ are added and multiplied in the usual way, with a bit of care. For example, $x^k + x^k = (1+1)x^k = 0x^k = 0$, so the degree of $f(x)+g(x)$ is not necessarily equal to the maximum of the degrees of $f(x)$ and $g(x)$. Thus for $f(x), g(x) \in K[x]$, we have $(f(x)+g(x))^2 = (f(x))^2 + (g(x))^2$.

**Definition 4.9 Division Algorithm** If $f(x), p(x) \in K[x]$ with $p(x) \neq 0$ then there exist unique polynomials

$$q(x), r(x) \in K[x] \text{ such that } f(x) = q(x)p(x) + r(x),$$

with $r(x) = 0$ or $\text{degree}(r(x)) < \text{degree}(p(x))$, and $r(x)$ is unique. As usual, $q(x)$ is called the *quotient* and $r(x)$ is called the *remainder*.

**Theorem 4.10** *Given a polynomial $p(x)$ of degree $n$, by associating $f(x)$ with the remainder $r(x)$ where*

$$f(x) = q(x)p(x) + r(x),$$

*each polynomial $f(x)$ corresponds to a unique polynomial of degree at most $n-1$, and hence to a unique word of length $n$.*

Thus, given any polynomial $f(x) \in K[x]$, we can represent $f(x)$ by a unique representative of degree less than $n$ from the set of residue (equivalence) classes of $K[x]$ modulo $p(x)$. We say that $f(x)$ equals $r(x)$ modulo $p(x)$, and write $f(x) = r(x) \mod p(x)$.
We can avoid polynomial long division when finding $r(x) \mod p(x)$ in the following way. If $p(x) = p_0 + p_1x + \ldots + p_{n-1}x^{n-1} + x^n$, then we can substitute for $x^n$ in $f(x)$ using

$$x^n = p_0 + p_1x + \ldots + p_{n-1}x^{n-1} \mod p(x).$$

**Definition 4.11** The polynomial $f(x) = \alpha_0 + \alpha_1x + \ldots + \alpha_{n-1}x^{n-1}$ of degree $n-1$ over $K$ may be regarded as the word $v = \alpha_0\alpha_1 \ldots \alpha_{n-1}$ of length $n$ in $K^n$.

If we fix the length $n$ of words we wish to consider, then every word of length $n$ corresponds to exactly one polynomial over $K$ of degree less than or equal to $n-1$. The word with all digits 0 is represented by the zero polynomial. Thus any code $C$ of length $n$ can be represented as a set of polynomials over $K$ of degree at most $n-1$.

**Definition 4.12** We shall use $p(x) = 1 + x^n$ to make the correspondence between polynomials and words of length $n$. Hence $x^n = 1 \pmod{p(x)}$, and the correspondence between polynomials and words of length $n$ is easily made.

## 4.3 Introduction to cyclic codes

Recall that if $v$ is a word of length $n$, then the *cyclic shift* of $v$, denoted $\gamma(v)$, is the word obtained by moving the last digit of $v$ to the beginning.

**Definition 4.13** A code $C$ is a *cyclic code* if $(v \in C)$ implies that $(\gamma(v) \in C)$.

**Example 4.14**

- $\{000, 110, 101, 011\}$ is both cyclic and linear.
- $\{000, 100, 011, 111\}$ is linear but not cyclic.
- $\{010, 100, 001\}$ is cyclic but not linear.

**Exercise 4.15** Find the smallest cyclic linear code containing the word 001.

Note that the cyclic shift preserves weight of words, so $wt(v) = wt(\gamma(v))$.

The Hamming codes are equivalent to cyclic codes. This is easy to prove for small Hamming codes, but is harder to prove in general.

We are using $p(x) = 1 + x^n$ to make the correspondence between polynomials and words of length $n$. The main reason for this is that (for cyclic codes) if the word $v$ is represented by the polynomial $f(x)$ then the word $\gamma(v)$ is represented by the polynomial $xf(x) \mod (x^n + 1)$.

**Example 4.16** Let $f(x) = 1 + x + x^5$ and use $p(x) = 1 + x^7$ to make the correspondence between polynomials and words of length 7. Then

$$\begin{aligned} f(x) = 1 + x + x^5 &\leftrightarrow 1100010 = w \\ xf(x) = x + x^2 + x^6 &\leftrightarrow 0110001 = \gamma(w) \\ x^2 f(x) = 1 + x^2 + x^3 &\leftrightarrow 1011000 = \gamma(\gamma(w)) \end{aligned}$$

**Definition 4.17** A set of polynomials of degree less than $n$ corresponds to a *linear* code if it is closed under addition. Now, we can say that a set of polynomials modulo $p(x) = 1 + x^n$ corresponds to a *cyclic* code if the set is closed under multiplication by $x$.

If we consider the set of polynomials mod $(x^n + 1)$, then the set of all polynomials of degree at most $n-1$ forms a ring with the usual definition of addition and multiplication (over $K$). Recall that a ring $(R, +, \cdot)$ is a set $R$ with two operations, addition and multiplication, such that $(R, +)$ is an abelian group and the ring has associative multiplication that is left and right distributive over addition. Thus, a ring $(R, +, \cdot)$ satisfies the following properties:

- Closure under addition: $\forall a, b \in R,\ a + b \in R$;
- Additive identity: $\exists\, 0 \in R$ such that $\forall a \in R,\ a + 0 = 0 + a = a$;
- Additive inverse: $\forall a \in R$, $\exists - a \in R$ such that $a + (-a) = (-a) + a = 0$;
- Associative addition: $\forall a, b, c \in R,\ a + (b + c) = (a + b) + c$;
- Commutative addition: $\forall a, b \in R,\ a + b = b + a$;
- Closure under multiplication: $\forall a, b \in R,\ a \cdot b \in R$;
- Associative multiplication: $\forall a, b, c \in R,\ a \cdot (b \cdot c) = (a \cdot b) \cdot c$;
- Distributivity: $\forall a, b, c \in R,\ a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ and $(a + b) \cdot c = (a \cdot c) + (b \cdot c)$.

**Definition 4.18** An *ideal* in a ring $R$ is a subset $I$ of $R$ such that

(i) $I$ is a subgroup of the additive group of $R$; and

(ii) for every $i \in I$ and $r \in R$ we have $ir \in I$ and $ri \in I$, so that $I$ is closed under multiplication by any element of $R$.

A cyclic linear code corresponds to an ideal in the ring of polynomial residues mod $(x^n + 1)$.

From now on, we shall refer to the set of polynomials corresponding to a code as the code itself and will use polynomial notation and usual bitwise notation interchangably.

**Definition 4.19** Let $C$ be a cyclic code of length $n$. A *generator* $g(x)$ for $C$ is a nonzero polynomial in $C$ of least degree.

**Exercise 4.20** Let $C$ be the cyclic linear code $C = \{0000, 1010, 0101, 1111\}$, so

$$C = \{0, 1 + x^2, x + x^3, 1 + x + x^2 + x^3\}.$$

By definition, $1 + x^2 = 1010$ is a generator for $C$. Show that every word in $C$ is a multiple mod $(1 + x^4)$ of this generator.

**Theorem 4.21** *Let $C \neq \{0\}$ be a cyclic linear code of length $n$. Then:*

1. *$C$ has a unique generator $g(x)$; and*
2. *every polynomial $f(x) \in C$ can be written in the form $f(x) = q(x)g(x)$ for a unique $q(x) \in K[x]$, where $\deg q(x) \leq n - 1 - \deg g(x)$.*

**Proof of (1):** $C$ is cyclic and contains at least one nonzero polynomial, so we can select from $C$ a nonzero polynomial $g(x)$ of smallest degree, say $k$. By definition, $g(x)$ is a generator for $C$.

Assume $C$ contains another polynomial $g_1(x)$ of degree $k$. Since $x^k + x^k = 0$, we have $g(x) + g_1(x)$ has degree less than $k$. But as $C$ is linear, $g(x) + g_1(x) \in C$. Thus, as $g(x)$ has minimal degree, we must have $g(x) + g_1(x) = 0$, so $g(x) = g_1(x)$ and $C$ has a unique generator.

**Proof of (2):** Now let $f(x)$ be any polynomial in $C$. Thus by the Division Algorithm there exist unique polynomials $q(x)$ and $r(x)$ in $K[x]$ such that

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

where $r(x) = 0$ or the degree of $r(x)$ is less than the degree of $g(x)$. Since $C$ is cyclic and linear, $q(x)g(x) \in C$. Since $C$ is linear and hence closed under addition, $r(x) = f(x) + q(x)g(x) \in C$. By the Division Algorithm, degree $r(x)$ is less than degree $g(x)$, and by minimality of the degree of the generator $g(x)$, it follows that $r(x) = 0$, so $f(x) = q(x)g(x)$. Thus $\deg f(x) = \deg q(x) + \deg g(x)$, and since $f(x) \in C$, we have $\deg f(x) \leq n - 1$. Hence $\deg q(x) \leq (n - 1) - \deg g(x)$. □

**Exercise 4.22** Find the smallest linear cyclic code $C$ of length 6 containing $x^2 + x^5$. Find a generator $g(x)$ for $C$, and represent each word in $C$ as a multiple of $g(x)$.

## 4.4 Generating matrices for linear cyclic codes

We now turn our attention to finding a generating matrix for a linear cyclic code $C$.

**Theorem 4.23** *If a linear cyclic code $C$ of length $n$ has generator $g(x)$ of degree $n-k$, then $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ forms a basis for $C$ and hence $\dim C = k$.*

**Proof:** First note that the polynomials $g(x), xg(x), \ldots, x^{k-1}g(x)$ are all distinct modulo $(1+x^n)$, since each has degree one greater than its predecessor and $x^{k-1}g(x)$ has degree $n-1$. To see that these polynomials form a basis for $C$, we must show that they both span $C$ and are linearly independent.

To see that they span $C$, let $f(x) \in C$. Then by Theorem 4.21, $f(x) = q(x)g(x)$ for some $q(x) \in K[x]$ where $\deg q(x) \leq n - 1 - \deg g(x)$ so $\deg q(x) \leq k-1$. Let

$$q(x) = q_0 + q_1 x + \ldots + q_l x^l, \text{ where } l \leq k-1$$

and $q_0, q_1, \ldots, q_l \in K$. Now $g(x)$ has degree $n-k$, so using $l = k-1$ (but noting that since $l \leq k-1$, we may have $q_{k-1} = 0$), we have

$$\begin{aligned} f(x) &= q(x)g(x) \\ &= (q_0 + q_1 x + \ldots + q_{k-1}x^{k-1})g(x) \\ &= q_0 g(x) + q_1 x g(x) + q_2 x^2 g(x) + \ldots + q_{k-1}x^{k-1}g(x). \end{aligned}$$

Thus any $f(x) \in C$ can be written as a linear combination of the elements of the set

$$\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$$

so this set spans $C$.

To see that the set of polynomials is linearly independent, let

$$g(x) = \alpha_0 + \alpha_1 x + \ldots + \alpha_{n-k-1}x^{n-k-1} + x^{n-k},$$

and associate with each polynomial in $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ the corresponding word of length $n$. Writing these words as the rows of a matrix gives

$$\begin{pmatrix} \alpha_0 & \alpha_1 & \alpha_2 & \ldots & \alpha_{n-k-1} & 1 & 0 & 0 & \ldots & 0 \\ 0 & \alpha_0 & \alpha_1 & \ldots & \alpha_{n-k-2} & \alpha_{n-k-1} & 1 & 0 & \ldots & 0 \\ 0 & 0 & \alpha_0 & \alpha_1 & \ldots & \alpha_{n-k-2} & \alpha_{n-k-1} & 1 & \ldots & 0 \\ \vdots & & & & \vdots & & & & & \vdots \\ 0 & 0 & 0 & \ldots & \alpha_0 & \alpha_1 & \ldots & \alpha_{n-k-2} & \alpha_{n-k-1} & 1 \end{pmatrix}.$$

It is clear from the form of the matrix (REF) that the rows are linearly independent and so $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ is a linearly independent set.
Thus $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ is a basis for $C$ and hence the dimension of $C$ is $k$. □

**Corollary 4.24** Let $C$ be a linear cyclic code of length $n$ with generator polynomial $g(x)$ of degree $n-k$. Then the $k \times n$ matrix

$$\mathbf{G} = \begin{bmatrix} g(x) \\ xg(x) \\ x^2g(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix}$$

is a generating matrix for $C$.

**Exercise 4.25** Let $C$ be the linear cyclic code of length $n = 7$, with generator $g(x) = 1 + x + x^3$. Find a generating matrix for $C$.

Note that the generating matrix in the previous exercise is a generating matrix for the $(7, 4, 3)$ Hamming code.

The previous exercise shows one benefit of using a cyclic code. Rather than storing the entire generating matrix for the code, it is only necessary to store the generator polynomial.

Let $C$ be a linear cyclic code of length $n$ and dimension $k$ with generator $g(x)$. The message word $a = a_0a_1 \ldots a_{k-1}$ of length $k$ to be encoded can be represented by the message polynomial $a(x) =$

$a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}$. Encoding using the generating matrix developed above then consists of polynomial multiplication; that is, $a(x)$ is encoded as $a(x)g(x)$ to give the codeword $c(x)$ of length $n$. The inverse operation to polynomial multiplication is polynomial division. Thus, finding the message $a(x)$ corresponding to the codeword $c(x)$ consists of dividing $c(x)$ by $g(x)$.

**Exercise 4.26** Consider the linear cyclic code $C$ from Exercise 4.25 with messages encoded using the procedure above. Determine the codeword corresponding to the message word 1010. Determine the message word corresponding to the codeword 1100101.

We can also find the generating matrix in standard form of a linear cyclic code.

**Theorem 4.27** *Let $g(x)$ be the generator of a linear cyclic code $C$ of length $n$. For $n-k \leq i \leq n-1$, let $r_i$ be the binary word of length $n-k$ corresponding to $r_i(x) = x^i \pmod{g(x)}$. Then a generating matrix for $C$ is:*

$$\mathbf{G} = \left[\begin{array}{c|c} \mathbf{I}_k & \begin{array}{c} r_{n-k} \\ \vdots \\ r_{n-1} \end{array} \end{array}\right]$$

**Proof:** By the Division Algorithm, $x^i = q(x)g(x) + r_i(x)$, so $x^i + r_i(x) = q(x)g(x)$. Thus $g(x)$ divides $x^i + r_i(x)$, so we have that $x^i + r_i(x)$ is a polynomial corresponding to a codeword in $C$. Since $\deg r_i(x) < \deg g(x)$ and $\deg x^i \geq \deg g(x)$, we have $x^i + r_i(x) \neq 0$ and $x^i + r_i(x) \in C$. Indeed, these words are linearly independent, as can be seen when they are listed in a matrix:

$$\left[\begin{array}{c|c} \begin{array}{c} r_{n-k} \\ \vdots \\ r_{n-1} \end{array} & \mathbf{I}_k \end{array}\right]$$

As this matrix has $k$ linearly independent rows, each of which is a codeword of $C$, it follows that it is a generating matrix for $C$. But $C$ is cyclic, so we can perform a cyclic shift $k$ times on this matrix. Hence a generating matrix for $C$ is:

$$\mathbf{G} = \left[\begin{array}{c|c} \mathbf{I}_k & \begin{array}{c} r_{n-k} \\ \vdots \\ r_{n-1} \end{array} \end{array}\right]$$

□

**Example 4.28** Find **G** in standard form for the linear cyclic code $C$ of length 7 generated by $g(x) = 1 + x + x^3$ (encountered in Exercise 4.25).

$$\begin{array}{ll} x^{n-k} = x^3 = 1 + x \pmod{g(x)} & r_3 = 110, \\ x^4 = x(1+x) = x + x^2 \pmod{g(x)} & r_4 = 011, \\ x^5 = x(x+x^2) = x^2 + x^3 = 1 + x + x^2 \pmod{g(x)} & r_5 = 111, \\ x^6 = x(1+x+x^2) = x + x^2 + x^3 = 1 + x^2 \pmod{g(x)} & r_6 = 101. \end{array}$$

Hence a generating matrix in standard form is:

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

## 4.5 Finding a generator polynomial for a linear cyclic code

The following theorem shows how to find all linear cyclic codes of a given length, by determining which polynomials are generators of a linear cyclic code of that length.

**Theorem 4.29** *The polynomial $g(x)$ is a generator for a linear cyclic code of length $n$ if and only if*

$$g(x) \mid (1 + x^n).$$

**Proof:** Assume that $C$ is a linear cyclic code of length $n$ with generator $g(x)$. By the Division Algorithm, there exist unique polynomials $q(x)$ and $r(x)$ such that

$$1 + x^n = q(x)g(x) + r(x),$$

where $r(x) = 0$ or $\deg r(x) < \deg g(x)$. Since algebra is done modulo $(1 + x^n)$ in $C$, we have $r(x) = q(x)g(x)$. Therefore $r(x) \in C$. By the minimality of the degree of the generator $g(x)$, we must have $r(x) = 0$. Thus $1 + x^n = q(x)g(x)$, so $g(x)|1 + x^n$.

Conversely, suppose that $g(x)|1 + x^n$. If $g(x) \neq 1 + x^n$, then $g(x)$ has degree $n - k$, for some $k$, $0 < k \leq n$. Thus $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$ is a linearly independent set (see Theorem 4.23). Define $C$ to be the span of $\{g(x), xg(x), \ldots, x^{k-1}g(x)\}$. Thus $C$ is a linear code. To see that $C$ is cyclic, let $f(x)$ be a linear combination of the above polynomials, so

$$f(x) = f_0 g(x) + f_1 xg(x) + f_2 x^2 g(x) + \cdots + f_{k-1} x^{k-1} g(x).$$

Then

$$xf(x) = f_0 xg(x) + f_1 x^2 g(x) + f_2 x^3 g(x) + \cdots + f_{k-2} x^{k-1} g(x) + f_{k-1} x^k g(x).$$

However,

$$x^k g(x) = (1 + x^n) + x^k g(x) = q(x)g(x) + x^k g(x) = (q(x) + x^k)g(x), \text{ for some } q(x)$$

since $g(x)$ is a factor of $1 + x^n$. Clearly,

$$\text{degree } q(x) = n - \text{ degree } g(x) = k,$$

so degree $(q(x) + x^k) \leq k - 1$. Therefore $f_{k-1} x^k g(x)$ is a linear combination of the polynomials in our basis for $C$ and so $xf(x) \in C$. Therefore, $C$ is cyclic. □

Thus, to find all linear cyclic codes of a given length $n$, find all factors of $1 + x^n$, and each such factor is a generator for a linear cyclic code of length $n$.

**Definition 4.30** The linear cyclic code of length $n$ generated by 1 is $K^n$, and the linear cyclic code of length $n$ generated by $1 + x^n$ is $\{\mathbf{0}\}$. These are called the *improper cyclic codes*. Other cyclic linear codes of length $n$ are called *proper cyclic codes*.

**Definition 4.31** A polynomial $p(x) \in K[x]$ is *irreducible* if $p(x) = q(x)s(x)$ implies either $q(x)$ or $s(x)$ has degree 0 (that is, the only factors of $p(x)$ are 1 and itself).

To find all factors of $1 + x^n$, we first find all irreducible factors of $1 + x^n$, then we form all possible products of these factors (except for the products 1 and $1+x^n$), then each such product is the generator for a proper linear cyclic code of length $n$.

Finding all irreducible factors of $1 + x^n$ is not necessarily easy. The task is simplified if we note the following: if the coefficients of a polynomial sum to 0, then $(x + 1)$ is a factor, and if the constant coefficient of a polynomial is 0, then $x$ is a factor. For reference, here is a table of all small-degree irreducible polynomials in $K[x]$.

| degree | irreducible polynomials |
|---|---|
| 1 | $x, \quad x + 1$ |
| 2 | $x^2 + x + 1$ |
| 3 | $x^3 + x^2 + 1, x^3 + x + 1$ |
| 4 | $x^4 + x^3 + x^2 + x + 1, \quad x^4 + x^3 + 1, \quad x^4 + x + 1$ |

**Exercise 4.32** Find all proper linear cyclic codes of length $n = 3$.

Note that if $n$ is even (so $n = 2y$), then

$$1 + x^n = 1 + x^{2y} = (1 + x^y)^2.$$

**Exercise 4.33** Determine the dimensions of all proper linear cyclic codes of length $n = 6$.

For reference, here are the factorisations of $1 + x^n$ into irreducible polynomials for some small and useful odd $n$:

$$\begin{aligned}
1+x &= 1+x \\
1+x^3 &= (1+x)(1+x+x^2) \\
1+x^5 &= (1+x)(1+x+x^2+x^3+x^4) \\
1+x^7 &= (1+x)(1+x+x^3)(1+x^2+x^3) \\
1+x^9 &= (1+x)(1+x+x^2)(1+x^3+x^6) \\
1+x^{15} &= (1+x)(1+x+x^2)(1+x+x^4)(1+x^3+x^4)(1+x+x^2+x^3+x^4) \\
1+x^{23} &= (1+x)(1+x+x^5+x^6+x^7+x^9+x^{11})(1+x^2+x^4+x^5+x^6+x^{10}+x^{11})
\end{aligned}$$

Note that the Golay code $C_{23}$ is equivalent to the linear cyclic code with generator $1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11}$.

**Exercise 4.34** Does there exist a linear cyclic code of length 15 and dimension 4?

**Exercise 4.35** Does there exist a linear cyclic code of length 15 having 4 codewords?

## 4.6 Error detection and correction using cyclic codes

Cyclic codes have a straightforward error detection and correction algorithm for correcting cyclic burst error patterns, so cyclic codes are normally used when the objective is to correct cyclic burst error patterns. Examples of such codes are Reed-Solomon codes (used in CDs).

**Definition 4.36** A definition of burst length in terms of polynomials is as follows.

- Suppose $e(x) = x^k e_1(x)$ for some $k$ and for some polynomial

  $$e_1(x) = 1 + \alpha_1 x + \alpha_2 x^2 + \cdots + \alpha_{m-1} x^{m-1} + x^m.$$

  Then the *burst length* of $e(x)$ is $\deg e_1(x) + 1 = m + 1$.

- $e(x)$ has *cyclic burst length* $l$ if the minimum degree of $x^k e_1(x) \bmod (1+x^n)$ for $k = 0, 1, \ldots n-1$ is $l - 1$.

**Example 4.37** For the word $e = 00010100$ we have the polynomial $e(x) = x^3 + x^5 = x^3(1 + x^2)$, so we have $k = 3$ and $e_1(x) = 1 + x^2$. Thus $e$ has burst length $2 + 1 = 3$.

The word $e = 1000110$ has polynomial form $e(x) = 1 + x^4 + x^5$ with $k = 0$ and $e_1(x) = 1 + x^4 + x^5$ so $e$ has burst length 6. However, $x^3(1 + x^4 + x^5) \mod (1 + x^7) = 1 + x + x^3$ (or equivalently 1101000) so $e$ has cyclic burst length 4.

As noted earlier, any linear cyclic code $C$ with generator polynomial $g(x)$ of degree $n - k$ has a generating matrix of the form:

$$\mathbf{G} = \left(\begin{array}{c|c} \mathbf{I}_k & \begin{array}{c} r_{n-k} \\ \vdots \\ r_{n-1} \end{array} \end{array}\right)$$

where $x^i \equiv r_i(x) \pmod{g(x)}$. Thus, by Algorithm 2.13, a parity check matrix is

$$\mathbf{H} = \begin{pmatrix} r_{n-k} \\ r_{n-k+1} \\ \vdots \\ r_{n-1} \\ \mathbf{I}_{n-k} \end{pmatrix}.$$

In Exercise 4.6 we showed that there may exist a linear code of length 15 and dimension 9 that is 3 cyclic burst error correcting (but not 3-error correcting). We now show that the linear cyclic code with generator $g(x) = 1 + x + x^2 + x^3 + x^6$ is such a code.

**Example 4.38** Let $C$ be the cyclic linear code of length 15 with generator

$$g(x) = 1 + x + x^2 + x^3 + x^6 = (1 + x + x^2)(1 + x^3 + x^4).$$

A parity check matrix for $C$ created using the above method is obtained as follows:

We have $n - k = 6$. Working modulo $g(x) = 1 + x + x^2 + x^3 + x^6$, we have:

$$\begin{array}{lll} x^6 \equiv & 1 + x + x^2 + x^3 & \rightarrow r_6 = 111100 \\ x^7 \equiv & x + x^2 + x^3 + x^4 & \rightarrow r_7 = 011110 \\ x^8 \equiv & x^2 + x^3 + x^4 + x^5 & \rightarrow r_8 = 001111 \\ x^9 \equiv & x^3 + x^4 + x^5 + (1 + x + x^2 + x^3) = & \\ & 1 + x + x^2 + x^4 + x^5 & \rightarrow r_9 = 111011 \end{array}$$

and so on. We thus obtain the parity check matrix:

$$\mathbf{H} = \left(\begin{array}{cccccc} 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 0 & 1 \\ \hline & & \mathbf{I}_6 & & & \end{array}\right)$$

**Claim:** The linear cyclic code of length 15 with generator $g(x) = 1 + x + x^2 + x^3 + x^6$ is a 3 cyclic burst error correcting code.

To prove this, we require all error patterns of length 15 with cyclic burst length at most 3 to have different syndromes. The syndrome is the sum of the rows of $\mathbf{H}$ corresponding to the locations of the ones in the error pattern. We require all these sums to be different.

The error pattern with cyclic burst length 0 belongs to the 0 coset (the code itself). Error patterns with cyclic burst length 1 have syndromes equal to rows of $\mathbf{H}$, all of which are different. Error patterns with cyclic burst length 2 have syndromes equal to sums of 2 (cyclically) adjacent rows of $\mathbf{H}$.
The error patterns with cyclic burst length 2 and their syndromes are:

| error pattern | syndrome | error pattern | syndrome |
|---|---|---|---|
| $1 + x$ | 100010 | $x^8 + x^9$ | 011001 |
| $x + x^2$ | 010001 | $x^9 + x^{10}$ | 110000 |
| $x^2 + x^3$ | 110100 | $x^{10} + x^{11}$ | 011000 |
| $x^3 + x^4$ | 011010 | $x^{11} + x^{12}$ | 001100 |
| $x^4 + x^5$ | 001101 | $x^{12} + x^{13}$ | 000110 |
| $x^5 + x^6$ | 111010 | $x^{13} + x^{14}$ | 000011 |
| $x^6 + x^7$ | 011101 | $1 + x^{14}$ | 111101 |
| $x^7 + x^8$ | 110010 | | |

Error patterns with cyclic burst length 3 have syndromes equal to either the sum of 3 (cyclically) adjacent rows of $\mathbf{H}$ or the sum of 2 rows of $\mathbf{H}$ which are the first and third rows in a set of 3 (cyclically) adjacent rows of $\mathbf{H}$.

The error patterns with cyclic burst length 3 and their syndromes are:

| error pattern | syndrome | error pattern | syndrome |
|---|---|---|---|
| $1 + x + x^2$ | 101101 | $1 + x^2$ | 110011 |
| $x + x^2 + x^3$ | 101010 | $x + x^3$ | 100101 |
| $x^2 + x^3 + x^4$ | 010101 | $x^2 + x^4$ | 101110 |
| $x^3 + x^4 + x^5$ | 110110 | $x^3 + x^5$ | 010111 |
| $x^4 + x^5 + x^6$ | 011011 | $x^4 + x^6$ | 110111 |
| $x^5 + x^6 + x^7$ | 110001 | $x^5 + x^7$ | 100111 |
| $x^6 + x^7 + x^8$ | 100100 | $x^6 + x^8$ | 101111 |
| $x^7 + x^8 + x^9$ | 010010 | $x^7 + x^9$ | 101011 |
| $x^8 + x^9 + x^{10}$ | 001001 | $x^8 + x^{10}$ | 101001 |
| $x^9 + x^{10} + x^{11}$ | 111000 | $x^9 + x^{11}$ | 101000 |
| $x^{10} + x^{11} + x^{12}$ | 011100 | $x^{10} + x^{12}$ | 010100 |
| $x^{11} + x^{12} + x^{13}$ | 001110 | $x^{11} + x^{13}$ | 001010 |
| $x^{12} + x^{13} + x^{14}$ | 000111 | $x^{12} + x^{14}$ | 000101 |
| $1 + x^{13} + x^{14}$ | 111111 | $1 + x^{13}$ | 111110 |
| $1 + x + x^{14}$ | 100011 | $x + x^{14}$ | 011111 |

All of these 61 syndromes are different. So $C$ is a 3 cyclic burst error correcting code.

As you can see from the above discussion, the creation and storage of an SDA can be costly. Luckily, the structure of cyclic linear codes allows us to decode received words containing cyclic burst errors

without the need for an SDA.
In this decoding process, we use the parity check matrix of the form

$$\mathbf{H} = \begin{pmatrix} r_{n-k} \\ r_{n-k+1} \\ \vdots \\ r_{n-1} \\ \mathbf{I}_{n-k} \end{pmatrix}.$$

For a received word $w$ with error pattern $e$, let $w(x)$ and $e(x)$ be the corresponding polynomials. For $0 \leq i \leq n-1$, let $w_i$ and $e_i$ be the words corresponding to $x^i w(x)$ and $x^i e(x) \pmod{1+x^n}$, respectively. Let $s = w\mathbf{H}$ be the syndrome of $w$, and let $s_i = w_i\mathbf{H}$ be the syndrome of $w_i$ for $0 \leq i \leq n-1$.

**Theorem 4.39** *Let $C$ be a $t$ cyclic burst error correcting linear cyclic code of length $n$ and dimension $k$ (so $t \leq n-k$). Let $w+e$ be a codeword, where $e \in C(B_n(t))$. Then there exists an $i$, $0 \leq i \leq n-1$ such that $s_i$ is a burst error pattern of length at most $t$ and so $e(x) = x^{n+k-i}s_i(x) \pmod{1+x^n}$.*

**Proof:** Let $w+e \in C$, where $e$ is a cyclic burst error pattern with burst of length at most $t$. Since $C$ is cyclic and $w+e \in C$, we have $w_i + e_i \in C$. Since $w_i + e_i \in C$, $w_i$ and $e_i$ are in the same coset of $C$ and hence

$$s_i = w_i\mathbf{H} = e_i\mathbf{H}.$$

Also, since $t \leq n-k$ (Lemma 4.7), it is clear that for some $i$, $0 \leq i \leq n-1$, the word $e_i$ has 0s in the first $k$ places. Now, since $s_i = e_i\mathbf{H}$ and $e_i$ has zeros in the first $k$ places, $e_i$ is of the form $e_i = [0, s_i]$. Since $e_i$ is a cyclic burst error pattern with burst of length at most $t$, then so is $s_i$.
Since $e_i = [0, s_i]$, we have $e_i(x) = x^k s_i(x)$. Thus $e(x) = x^{n-i}e_i(x) = x^{k+n-i}s_i(x)$. □

Theorem 4.39 shows how to decode a received word $w$: calculate the syndromes of the cyclic shifts of $w$ until a syndrome $s_i$ is found which is a burst error pattern with burst of length at most $t$; then $n-i$ cyclic shifts of $e_i = (0, s_i)$ will result in the most likely cyclic burst error pattern $e$.
The drawback to this method is the number of syndromes that need to be calculated. The next lemma gives us a way to simplify this process, by showing how to calculate $s_{i+1}$ from $s_i$.

**Lemma 4.40** *If $s$ is the syndrome of $w$ with corresponding polynomial $s(x)$, then $s_1(x) = xs(x) \pmod{g(x)}$ is the polynomial corresponding to the syndrome of $w_1$.*

**Proof:** For $0 \leq i \leq n-1$, let $h_i$ be the binary word in the $i$th row of $\mathbf{H}$ and let $h_i(x)$ be its polynomial representation. Then $h_{i+1}(x) = xh_i(x) \pmod{g(x)}$. Let $w(x) = \sum_{i=0}^{n-1} a_i x^i$ be a received word. Then the syndrome of $w$ is $s(x) = \sum_{i=0}^{n-1} a_i h_i(x)$. Then $w_1(x) = \sum_{i=0}^{n-1} a_i x^{i+1}$ and so

$$s_1(x) = \sum_{i=0}^{n-1} a_i h_{i+1}(x) = x\sum_{i=0}^{n-1} a_i h_i(x) = xs(x) \pmod{g(x)}.$$

□

**Example 4.41** Let $C$ be the 4 cyclic burst error correcting linear cyclic code of length 15 and dimension 7 with generator

$$g(x) = 1 + x^4 + x^6 + x^7 + x^8.$$

Find a parity check matrix for $C$ and use it to decode the received word

$$w = 011\ 100\ 011\ 001\ 110.$$

We first need to calculate $r_8, r_9, \ldots, r_{14}$.

$$\begin{aligned}
x^8 &= 1 + x^4 + x^6 + x^7 \leftrightarrow 10001011 \\
x^9 &= 1 + x + x^4 + x^5 + x^6 \leftrightarrow 11001110 \\
x^{10} &= x + x^2 + x^5 + x^6 + x^7 \leftrightarrow 01100111 \\
x^{11} &= 1 + x^2 + x^3 + x^4 \leftrightarrow 10111000 \\
x^{12} &= x + x^3 + x^4 + x^5 \leftrightarrow 01011100 \\
x^{13} &= x^2 + x^4 + x^5 + x^6 \leftrightarrow 00101110 \\
x^{14} &= x^3 + x^5 + x^6 + x^7 \leftrightarrow 00010111
\end{aligned}$$

Thus we find:

$$\mathbf{H} = \begin{pmatrix}
1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\
1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 \\
0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 1 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \\
 & & & & \mathbf{I}_8 & & &
\end{pmatrix}$$

To decode the received word $w = 011100011001110$, we first calculate the syndrome of $w$:

$$s = w\mathbf{H} = 11011111, \text{ so } s(x) = 1 + x + x^3 + x^4 + x^5 + x^6 + x^7.$$

Then we apply Lemma 4.40 to calculate $s_i$ for $i = 1, 2, \ldots$ until we find a syndrome that contains a burst error pattern with burst length at most 4.

$$\begin{aligned}
&s_1(x) = xs(x) = x + x^2 + x^4 + x^5 + x^6 + x^7 + (1 + x^4 + x^6 + x^7) = 1 + x + x^2 + x^5 && \text{so } s_1 = 11100100 \\
&s_2(x) = xs_1(x) = x + x^2 + x^3 + x^6 && \text{so } s_2 = 01110010 \\
&s_3(x) = xs_2(x) = x^2 + x^3 + x^4 + x^7 && \text{so } s_3 = 00111001 \\
&s_4(x) = xs_3(x) = x^3 + x^4 + x^5 + (1 + x^4 + x^6 + x^7) = 1 + x^3 + x^5 + x^6 + x^7 && \text{so } s_4 = 10010111 \\
&s_5(x) = xs_4(x) = x + x^4 + x^6 + x^7 + (1 + x^4 + x^6 + x^7) = 1 + x && \text{so } s_5 = 11000000
\end{aligned}$$

Since $s_5$ is a burst error pattern with burst of length 2 (less than 4) we have $e_5 = 000000011000000$ and hence $e = 001100000000000$. Thus the most likely transmitted codeword is

$$w + e = 010000011001110.$$

If we are applying this decoding process with a $t$ cyclic burst error correcting code and we do not obtain a syndrome with burst length at most $t$, then we conclude that a cyclic burst of more than $t$ errors has occurred and we request retransmission (if possible).

**Exercise 4.42** Use the code $C$ with generator $g(x)$ from Example 4.41 to decode the received word $w = 110000100000101$.

## 4.7 Another parity check matrix for a linear cyclic code

To find a parity check matrix for a linear cyclic code, we could apply Algorithm 2.13 to the RREF generating matrix for the code and obtain the parity check matrix we have been using in the previous section. However, it is sometimes useful to to have the following method for finding a parity check matrix for a linear cyclic code.

**Definition 4.43** If $g(x)$ is a generating polynomial for a linear cyclic code $C$ of length $n$ and $h(x)$ is a polynomial such that $g(x)h(x) = x^n + 1$, then we call $h(x)$ a *parity check polynomial* for $C$.

**Theorem 4.44** *Let $g(x)$ be a generating polynomial for a linear cyclic code $C$ of length $n$ and let $h(x)$ be a polynomial such that $g(x)h(x) = x^n + 1$. Then:*

1. *$f(x) \in C$ if and only if $f(x)h(x) = 0$*
2. *The matrix, the reversal of whose columns are*

$$h(x), xh(x), \ldots, x^{n-k-1}h(x)$$

*(where $\dim C = k$), is a parity check matrix for $C$.*

**Example 4.45** Let $C$ be the linear cyclic code of length 7 generated by $g(x) = 1 + x + x^2 + x^4$. Then $1 + x^7 = g(x)h(x)$ where $h(x) = 1 + x + x^3$.

$$\begin{aligned} h(x) &= 1 + x + x^3 \\ xh(x) &= x + x^2 + x^4 \\ x^2h(x) &= x^2 + x^3 + x^5 \\ x^3h(x) &= x^3 + x^4 + x^6 \end{aligned}$$

We write the corresponding binary words as columns (from bottom to top) to obtain a parity check matrix for $C$. Thus a generating matrix and parity check matrix for $C$ are given by:

$$\mathbf{G} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{pmatrix} \qquad \text{and} \qquad \mathbf{H} = \begin{pmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}.$$

## 4.8 Interleaving

One method for improving the burst error correcting capability of a code is to make use of interleaving. This technique rearranges the order in which the digits of the codewords are transmitted.

**Definition 4.46** Codewords are said to be *interleaved to depth $s$* if the digits of the codewords $c_1, c_2, \ldots, c_s$, where

$$\begin{aligned} c_1 &= (c_{1,1},\ c_{1,2},\ \ldots,\ c_{1,n}) \\ c_2 &= (c_{2,1},\ c_{2,2},\ \ldots,\ c_{2,n}) \\ &\vdots \qquad\quad \vdots \\ c_s &= (c_{s,1},\ c_{s,2},\ \ldots,\ c_{s,n}) \end{aligned}$$

are transmitted in the order

$$c_{1,1}\ c_{2,1}\ \ldots\ c_{s,1}\ c_{1,2}\ c_{2,2}\ \ldots\ c_{s,2}\ c_{1,3}\ c_{2,3}\ \ldots\ c_{s,n-1}\ c_{1,n}\ c_{2,n}\ \ldots\ c_{s,n}.$$

**Exercise 4.47** Let $C$ be the linear code with generating matrix $\mathbf{G} = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$. Write down the string of digits to be transmitted if the following codewords were interleaved to depth 3.

$$c_1 = 100110 \quad c_2 = 010101 \quad c_3 = 111000 \quad c_4 = 010101 \quad c_5 = 100110 \quad c_6 = 111000$$

**Theorem 4.48** Let $C$ be an $l$ burst error correcting code. If $C$ is interleaved to depth $s$ then all bursts of length at most $sl$ will be corrected, providing that each codeword is affected by at most one burst of errors.

**Example 4.49** The code $C$ in Exercise 4.47 is a 1-error correcting code. When interleaved to depth 3 it corrects all bursts of length 3 (provided those bursts are sufficiently separated).

The provision in Theorem 4.48 that each codeword is affected by at most one burst of errors requires that bursts of errors are separated by sufficiently long periods of error free transmission. Choosing $s$ to be large increases the burst length that can be corrected, but also increases the required length of error free transmission. Thus a good choice of $s$ will depend on some knowledge of the likelihood of periods of error free transmission on the channel.

In practice, the encoding of a message often uses two codes. For example, two Reed-Solomon codes are used in the encoding of music onto compact discs, and two codes are used by NASA and the European Space Agency in currect space communications where one code is a Reed-Solomon code and the other is a convolutional code. Interleaving to depth $s$ is an important technique in this two step encoding process.

Let $C_1$ be an $(n_1, k_1, \delta_1)$ linear code and let $C_2$ be an $(n_2, k_2, \delta_2)$ linear code. *Cross-interleaving* of $C_1$ with $C_2$ is done as follows. Messages (of length $k_1$) are first encoded using $C_1$ and the resulting codewords (of length $n_1$) are interleaved to depth $k_2$. Taking the sequence of consecutive digits arising from a particular position in a set of $k_2$ codewords gives a word of length $k_2$. These words are then regarded as messages and encoded using $C_2$. The codewords (of length $n_2$) can themselves be interleaved to any appropriate depth $s$ before transmission.

**Example 4.50** Let $C_1$ and $C_2$ be the codes with generating matrices

$$\mathbf{G}_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix} \quad \text{and} \quad \mathbf{G}_2 = \begin{pmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{pmatrix}$$

respectively. Then $(n_1, k_1, \delta_1) = (8, 4, 4)$ and $(n_2, k_2, \delta_2) = (6, 3, 3)$.
We shall encode the message words $m_1 = 1000$, $m_2 = 1100$ and $m_3 = 1010$ by the cross-interleaving of $C_1$ with $C_2$, and also interleave codewords of $C_2$ to depth $s = 3 = \delta_1 - 1$. Encoding $m_1$, $m_2$ and $m_3$ with $C_1$ gives

$$c_1 = m_1\mathbf{G}_1 = 10001110, \quad c_2 = m_2\mathbf{G}_1 = 11000011 \quad \text{and} \quad c_3 = m_3\mathbf{G}_1 = 10100101.$$

If we interleave these to depth $k_2 = 3$ and take the digits as message words of length 3, we obtain the following messages:

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

These are encoded using $C_2$ to produce eight codewords which are then interleaved to depth $s = 3$:

$$\begin{array}{lll} c_1' = 111000 & c_4' = 000000 & c_7' = 110011 \\ c_2' = 010101 & c_5' = 100110 & c_8' = 011110 \\ c_3' = 001011 & c_6' = 101101 & \end{array}$$

(Note that $c_7'$ and $c_8'$ will be interleaved with the first codeword $c_9'$ produced from the next three messages $m_4$, $m_5$ and $m_6$). So the string of digits to be transmitted begins

$$100110101010001011011000001011010001\ldots$$

The advantage of this two step encoding process is that $C_2$ can be used to detect $\delta_2 - 1$ errors, rather than to correct $\lfloor(\delta_2 - 1)/2\rfloor$ errors. From a string of received digits, once we undo the final interleaving from above, then we have received words of length $n_2$ to be compared to codewords of $C_2$. If errors are detected in such a received word, then all digits in this codeword are flagged and treated as digits that may be incorrect. Then we undo the cross-interleaving and consider the resulting words of length $n_1$ using the code $C_1$. Notice that if we know that $n_1 - \delta_1 + 1$ digits in a received word of length $n_1$ are correct (unflagged), then we can find the remaining $\delta_1 - 1$ digits, and identify the most likely codeword from $C_1$ and hence the most likely message word.

**Example 4.51** Suppose we receive the transmission from Example 4.50 above but the first 6 digits are transmitted incorrectly, so we receive

$$011001101010001011011000001011010001101111011010110100\ldots$$

The decoding process to determine the most likely first three message words would proceed as follows.

We write the string of received digits in three ($s$) rows by writing column 1 first, then column 2 etc... The rows then give us received words for the code $C_2$.

$$\begin{array}{lllllllll} c_a & = & 001000 & c_d & = & 000000 & c_g & = & 110011 \\ c_b & = & 100101 & c_e & = & 100110 & c_h & = & 011110 \\ c_c & = & 111011 & c_f & = & 101101 & c_i & = & 111000 \end{array}$$

We now use the parity check matrix $\mathbf{H}$ to determine which of the words above are codewords of $C_2$.

$$\mathbf{H} = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \qquad \begin{array}{lll} c_a\mathbf{H} = 011, & c_d\mathbf{H} = 000, & c_g\mathbf{H} = 000, \\ c_b\mathbf{H} = 011, & c_e\mathbf{H} = 000, & c_h\mathbf{H} = 000, \\ c_c\mathbf{H} = 011, & c_f\mathbf{H} = 000, & c_i\mathbf{H} = 000. \end{array}$$

Thus the first three are not codewords but the next six are codewords. We flag all 18 digits of the first three words.
Since the generating matrix for $C_2$ is in standard form, we obtain the following message words

$$\begin{array}{lll} c_a \to *** & c_d \to 000 & c_g \to 110 \\ c_b \to *** & c_e \to 100 & c_h \to 011 \\ c_c \to *** & c_f \to 101 & c_i \to 111 \end{array}$$

We write these message words in columns, so that the three ($k_2$) rows give us the received words for the code $C_1$.

$$\begin{array}{lllll} c_1 & = & ***01110 & \text{which corresponds to the codeword} & 10001110 \\ c_2 & = & ***00011 & \text{which corresponds to the codeword} & 11000011 \\ c_3 & = & ***00101 & \text{which corresponds to the codeword} & 10100101 \end{array}$$

Again, since the generating matrix for $C_1$ is in standard form, we can determine that the most likely first three message words were

$$m_1 = 1000 \qquad m_2 = 1100 \qquad m_3 = 1010.$$