Week 1

Introduction. How to detect and correct errors?

Version 2023-09-30. To accessible online version of this chapter

These notes are being revised to reflect the content of the course as taught in the 2023/24 academic year. Questions and comments on these lecture notes should be directed to `Yuri.Bazlov@manchester.ac.uk`.

Synopsis

We discuss information transmission and introduce the most basic notions of Coding Theory: **channel**, **alphabet**, **symbol**, **word**, **Hamming distance** *and of course* **code**. *We show how a code can detect and correct some errors that occur during transmission. We illustrate the process using two simple examples of codes.*

What is information?

It is fair to say that our age is the age of information. Huge quantities of information and data literally flow around us and are stored in various forms.

Information processing gives rise to many mathematical questions. Information needs to be processed because we may need, for example, to:

- *store* the information;
- *encrypt* the information;
- *transmit* the information.

For practical purposes, information needs to be stored efficiently, which leads to problems such as *compacting* or *compressing* the information. For the purposes of data protection and security, information may need to be *encrypted*. We will NOT consider these problems here. The course basically addresses one (extremely important) problem that arises in connection with *information transmission*.

We do not attempt to give an exhaustive definition of *information*. Whereas some mathematical models for space, time, motion were developed hundreds of years ago, the modern mathematical theory of information was only born in 1948 in the paper *A Mathematical Theory of Communication* by **Claude Shannon** (1916–2001). The following will be enough for our purposes:

Definition: information, alphabet, symbol

Fix a finite set F of two or more elements and call it **the alphabet**.
Elements of F are called **symbols**.
By **information** we mean a stream (a sequence) of symbols.

What does it mean to transmit information? What is a channel?

Informally, it means that symbols are sent by one party (the sender) and are received by another party (receiver). The symbols are transmitted via some medium, which we will in general refer to as the channel. More precisely, the channel is a mathematical abstraction of various real-life media such as a telephone line, a satellite communication link, a voice (in a face to face conversation between individuals), a CD (the sender writes information into it — the user reads the information from it), etc.

In this course we will assume that when a symbol is fed into the channel (the input symbol), the same or another symbol is read from the other end of the channel (the output symbol). Thus, we will only consider channels where neither erasures (when the output symbol is unreadable) nor deletions (when some symbols fed into the channel simply disappear) occur. Working with those more general channels requires more advanced mathematical apparatus which is beyond this course.

Importantly, we assume that there is *noise* in the channel, which means that the symbols are randomly changed by the channel. Our **model of information transmission** is thus as follows:

$$\boxed{\text{sender}} \xrightarrow{\text{stream of symbols}} \boxed{\text{channel}} \xrightarrow[\text{with random changes}]{\text{stream of symbols}} \boxed{\text{receiver}}$$

The above discussion leads us to the following simplified definition of a channel, which will be sufficient for this course.

Definition: memoryless channel

Fix the alphabet F. A **memoryless channel** is a function which has one argument — the input symbol $x \in F$ — and one output, a **random** output symbol $y \in F$ which depends only on x.

The definition says for each $x \in F$ which is sent via the channel (i.e., given as input), there is a probability distribution on F which shows which symbol will be received (output) with which probability.

A *noiseless* channel is the identity function which reads the input symbol x and outputs x with probability 1. However, more generally a channel is *noisy*, meaning that, if x is sent, the received symbol can be either x with probability less than 1, or another symbol.

Memoryless means that the received symbol depends only on x and not on symbols which may have been sent via the channel before x. We describe an example of a memoryless noisy channel, $BSC(p)$, below.

When a symbol $x \in F$ is sent, there are two possible outcomes:

- The received symbol is x. We say that no error occurred in this symbol.
- The received symbol is $y \neq x$. An error occurred in this symbol.

We formalise this in

Definition: symbol error

A **symbol error** is an event where a symbol $x \in F$ is sent via the channel, and the received symbol is y such that $y \neq x$.

The binary symmetric channel with bit error rate p

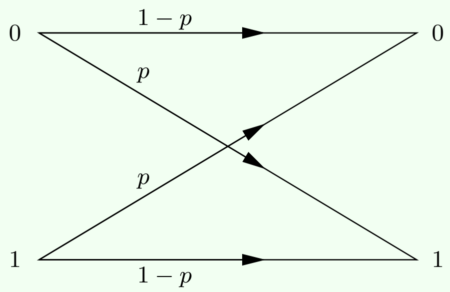
Our most basic example of a channel "speaks" the binary alphabet, which we will now define.

Definition: binary alphabet, bit

The **binary alphabet** is the set $\mathbb{F}_2 = \{0, 1\}$.
A **bit** (the same as **binary symbol**) is an element of $\mathbb{F}_2$.

Definition: $BSC(p)$

The **binary symmetric channel with bit error rate** p transmits binary symbols according to the following rule. A bit (0 or 1), sent via the channel, is received unchanged with probability $1-p$, and gets flipped with probability p:



Theoretically, we can consider $BSC(p)$ with $0 \le p < \frac{1}{2}$. Real-world binary channels are modelled by $BSC(p)$ with small p ranging from 10^{-2} (modem over a bad copper telephone line in 1950s) down to 10^{-13} (modern fibre optics).

There are other channels which are mathematical models of media not well-approximated by $BSC(p)$. This includes channels which "speak" alphabets other than $\mathbb{F}_2$. Much of theory we develop will work for general channels. However, explicit caculation of probabilities will only be done for $BSC(p)$.

What is a code and what is it used for?

A word is a finite sequence of symbols, and a code is a set of words. However, in this course we only consider *block codes* — this means that all the words in the code are of the same length n. Although variable length codes are used in modern applications, they are beyond the scope of the course, and so we refer to block codes simply as *codes*.

The main application of codes is **channel coding.** This means that the sender uses the chosen code to **encode** information before sending it via the channel. This allows the receiver to achieve one of the following goals:

- **detect** most errors that occur in the channel and ask the sender to retransmit the parts where errors are detected; or
- **correct** most errors that occur in the channel (nothing is retransmitted).

We will now formalise the definition of a code, explain encoding, and then show how error detection works and how error correction works.

Definition: word

A **word of length** n in the alphabet F is an element of F^n. Here F^n is the set of all n-tuples of symbols:

$$F^n = \{\underline{v} = (v_1, v_2, \dots, v_n) \mid v_i \in F,\ 1 \leq i \leq n\}.$$

Notation: words

We may write a word $(w_1, w_2, \dots, w_n) \in F^n$ as $w_1 w_2 \dots w_n$ if this is unambiguous. So, for example, the binary words 000, 101 and 111 belong to $\mathbb{F}_2^3$, and are more fully written as $(0,0,0)$, $(1,0,1)$ and $(1,1,1)$, respectively.
We will denote words by underlined letters: $\underline{w}$. Thus, $\underline{w} = w_1 w_2 \dots w_n$ where w_i denotes the ith symbol of the word $\underline{w}$.

Definition: code, codeword

A **code** of length n in the alphabet F is a non-empty subset of F^n. We will denote a code by C. That is, $C \subseteq F^n$, $C \neq \varnothing$.
A **codeword** is an element of the code.

The sender and the receiver choose a code $C \subseteq F^n$ for channel coding. To perform error detection and correction, the sender **must send only codewords** via the channel. However, information may contain arbitrary sequences of symbols, not just codewords, and so needs to be **encoded.**

The encoding procedure that we consider requires the code C to have the same number of elements as F^k, the set of words of length k, for some positive integers k. Note that $C \subseteq F^n$ and $\#C = \#(F^k)$ means that $k \leq n$.

Definition: encoder

An encoder for a code C is a bijective function $\texttt{ENCODE}\colon F^k \to C$.

Here is what the sender must do.

Procedure: encoding

Before transmission, a code C and an encoder $\texttt{ENCODE}\colon F^k \to C$ must be fixed.

- The information stream is split up into chunks of length k, called **messages**.
- The sender takes each message $\underline{u} \in F^k$ and replaces it with the codeword $\underline{c} = \texttt{ENCODE}(\underline{u})$.
- Each codeword $\underline{c}$ is sent into the channel.

Note: "$\underline{c}$ sent into the channel" means that the symbols $c_1, c_2, \dots, c_n$ are consecutively sent via the channel.

How to use a code to detect errors?

Recall that the sender transmits **only codewords:**

$$\boxed{\text{sender}} \xrightarrow[\underline{c}\in C]{\text{codeword}} \boxed{\text{channel}} \xrightarrow[\underline{y}\in F^n]{\text{received word}} \boxed{\text{receiver}}$$

The sender transmits a codeword $\underline{c} \in C$. The receiver receives a word $\underline{y} = y_1 y_2 \dots y_n$ which may not be not the same as $\underline{c}$, due to noise in the channel. Of course, if $\underline{y} \notin C$, the receiver knows that $\underline{y}$ is not what was sent.

If, however, $\underline{y} \in C$, the receiver has no way of knowing whether an error occurred, and must assume that there was no error. The above suggests the following

Procedure: error detection

1. The sender sends a codeword $\underline{c} \in C$ via the channel.
2. The receiver receives a word $\underline{y} \in F^n$:
   - if $\underline{y} \notin C$, this is a **detected error,** and the receiver asks the sender to retransmit the current codeword;
   - if $\underline{y} \in C$, the receiver accepts $\underline{y}$. If $\underline{y} \in C$ and $\underline{y} \neq \underline{c}$, this is an **undetected error.**
3. If there are any more codewords to transmit, return to step 1.

We assume that every detected error is eliminated by retransmitting the codeword one or more times. These retransmissions slow down the communication, but eventually the information accepted by the receiver will consist of correct codewords and codewords containing an undetected error.

How good is a code at detecting errors?

To select the most suitable error-detecting code for a particular application, or to decide whether to use a code at all, we need to measure how good is error detection.

One way to quantify this is to ask how many symbol errors must occur in a codeword to result in an undetected error. We will investigate this next week. This approach does not explicitly take the parameters of the channel into account.

Another approach is to calculate $P_{\text{undetect}}(C)$ for a particular channel:

Definition: $P_{\text{undetect}}(C)$, the probability of an undetected error

Suppose that an alphabet F and a channel are given. Let $C \subseteq F^n$ be a code. Assume that a random codeword from C is sent via the channel. Then $P_{\text{undetect}}(C)$ is the probability that an undetected error occurs in the received word.

Note that $P_{\text{undetect}}(C)$ expresses the average proportion of wrong codewords accepted by the receiver. Good error detection means that $P_{\text{undetect}}(C)$ is low.

E_3: an example of a code used for error detection

We now introduce the code E_3. Later, it will be seen as a particular case of E_n, the *binary even weight code of length n.* The code consists of four codewords:

Definition: the code E_3

$E_3 = \{000, 011, 101, 110\}$, a subset of $\mathbb{F}_2^3$.

To set up error detection based on E_3, we need an encoder. A standard choice is as follows: E_3 consists exactly of the binary words of length 3 which have an even number of 1s. Hence a 2-bit message can be encoded into a 3-bit codeword of E_3 by appending 0 or 1 so as to make the total number of 1s even. The appended bit is known as the **parity check bit:**

Example: encoder for E_3 (appending the parity check bit)

Define `ENCODE`: $\mathbb{F}_2^2 \to E_3$ by $00 \mapsto 000$, $01 \mapsto 011$, $10 \mapsto 101$, $11 \mapsto 110$.

Thus, if the information which needs to be transmitted is 001011, it is broken up into messages 00, 10, 11, then E_3-encoded as codewords 000, 101, 110 and sent via the channel.

How good is E_3 at detecting errors? E_3 is a binary code, so we may assume that the communication channel is $BSC(p)$ and calculate $P_{\text{undetect}}(E_3)$.

Suppose that the codeword $000 \in E_3$ is sent. For each word $\underline{y} \in \mathbb{F}_2^3$ we find the probability that $\underline{y}$ is received:

- 000 is received with probability $(1-p)^3$ — for each of the three bits, the probability of arriving unchanged is $1-p$;
- 001, 010, 100 have probability $p(1-p)^2$ each, and are detected errors;
- 011, 101 and 110, with probability $p^2(1-p)$ each, are undetected errors;
- 111 has probability p^3 to be received, and is a detected error.

The total probability of an undetected error is $3p^2(1-p)$.

If any codeword other than 000 is sent, the probability of an undetected error is the same (**Exercise:** check this).

Hence $P_{\text{undetect}}(E_3) = 3p^2(1-p)$, assuming the channel is $BSC(p)$. Later in the course, we will obtain a formula for $P_{\text{undetect}}(C)$ for a large class of binary codes C to avoid doing a case-by-case analysis each time.

Thus, if information is sent unencoded (E_3 is not used, no error detection), the average proportion of incorrect bits in the output will be p. If, however, E_3 is used for error detection, the average proportion of incorrect bits in the output will be less than $3p^2$. If $p \ll 1$, then $3p^2$ is much less than p — communication becomes more reliable when an error-detecting code is used.

How to use a code for error correction?

In order to take full advantage of error detection, the receiver should be able to contact the sender to request retransmission. In some situations this is not possible. We will now see how to modify the above error detection set-up so that the receiver could recover from errors, without contacting the sender.

Mathematics behind error correction got to be associated with the name of **Richard Hamming** (1915–1998) who came up with an idea to set up an efficient **error-correcting code.** In engineering literature, the set-up we are going to describe is referred to as **forward error correction (FEC).**

The first basic concept we need for error correction is distance between words.

Definition: Hamming distance

The **Hamming distance** between two words $\underline{x}, \underline{y} \in F^n$ is the number of positions where the symbol in $\underline{x}$ differs from the symbol in $\underline{y}$:

$$d(\underline{x}, \underline{y}) = \#\{i \in \{1, \dots, n\} : x_i \neq y_i\}.$$

Example: Hamming distace between some pairs of binary words

For example, in the set $\mathbb{F}_2^3$ of 3-bit binary words one has

$$d(101, 111) = 1 \qquad \text{and} \qquad d(101, 000) = 2.$$

Of course,

$$d(101, 101) = 0.$$

Lemma 1.1: properties of the Hamming distance

For any words $\underline{x}, \underline{y}, \underline{z} \in F^n$,

1. $d(\underline{x}, \underline{y}) \geq 0$; $d(\underline{x}, \underline{y}) = 0$ iff $\underline{x} = \underline{y}$.
2. $d(\underline{x}, \underline{y}) = d(\underline{y}, \underline{x})$.
3. $d(\underline{x}, \underline{z}) \leq d(\underline{x}, \underline{y}) + d(\underline{y}, \underline{z})$ (**the triangle inequality**).

Remark: *recall that a function $d(-,-)$ of two arguments which satisfies axioms 1.–3. is called a* metric. *This is familiar to those who studied* Metric spaces. *The Lemma says that the Hamming distance turns F^n into a metric space.*

Proof. 1. Since $d(\underline{x}, \underline{y})$ is a cardinality of a subset of $\{1, \ldots, n\}$, it is an integer between 0 and n. Moreover, $d(\underline{x}, \underline{y})$ is 0 iff $x_i = y_i$ for all i meaning that $\underline{x} = \underline{y}$.

2. Symmetry is clear as $x_i \neq y_i$ is equivalent to $y_i \neq x_i$.

3. An index i such that $x_i = y_i$ and $y_i = z_i$ does not contribute to $d(\underline{x}, \underline{y})$ nor to $d(\underline{y}, \underline{z})$ nor to $d(\underline{x}, \underline{z})$ (because $x_i = z_i$).

An index i such that $x_i \neq y_i$ or $y_i \neq z_i$ contributes at least 1 to $d(\underline{x}, \underline{y}) + d(\underline{y}, \underline{z})$, and can contribute at most 1 to $d(\underline{x}, \underline{z})$.

Thus, every index i contributes to left-hand side at most as much as to the right-hand side. Summing for i running over $\{1, \ldots, n\}$ proves the triangle inequality. ☐

We will now use the Hamming distance to set up error correction. Let $C \subseteq F^n$ be a code.

Definition: decoder, nearest neighbour

A **decoder** for C is a function $\mathtt{DECODE}\colon F^n \to C$ such that for any $\underline{y} \in F^n$, $\mathtt{DECODE}(\underline{y})$ is a nearest neighbour of $\underline{y}$ in C.
A **nearest neighbour** of $\underline{y} \in F^n$ in C is a codeword $\underline{c} \in C$ such that

$$d(\underline{c}, \underline{y}) = \min\{d(\underline{z}, \underline{y}) : \underline{z} \in C\}.$$

In order to use error correction, the sender and the receiver choose a decoder $\mathtt{DECODE}\colon F^n \to C$. The sender transmits codewords of C. The receiver **decodes** the received words:

$$\boxed{\text{sender}} \xrightarrow[\underline{c} \in C]{\text{codeword}} \boxed{\text{channel}} \xrightarrow[\underline{y} \in F^n]{\text{received word}} \xrightarrow[\mathtt{DECODE}(\underline{y}) \in C]{\text{decoded word}} \boxed{\text{receiver}}$$

Remark: what the decoder does; unencode

Thus, if the received word $\underline{y}$ is not a codeword, the decoder assumes that the codeword **closest** to $\underline{y}$ was sent, and outputs such a codeword.
To restore the original message $\underline{u} \in F^k$, the codeword $\underline{c} \in C$ can be **unencoded:** $\underline{u} = \texttt{ENCODE}^{-1}(\underline{c})$.

It may happen that some words $\underline{y} \in F^n$ have more than one nearest neighbour in C, which means that there exist more than one decoder function. In this course we assume that the receiver fixes one particular decoder to make decoding deterministic.

Definition: decoded correctly

In the above setup, let $\underline{c} \in C$ be the transmitted codeword and let $\underline{y} \in F^n$ be the received word. If $\texttt{DECODE}(\underline{y}) = \underline{c}$, we say that the received word is **decoded correctly.**

The following Claim shows that the error-correcting setup makes sense — at least, if *no* symbol errors occurred in a codeword, the decoder will not introduce errors! Strictly speaking, the Claim is unnecessary, because it will follow from part 2 of Theorem 2.1 given later.

Claim: a codeword is always decoded to itself

A codeword is its own unique nearest neighbour: indeed, $d(-,-)$ is non-negative hence $d(\underline{c}, \underline{c}) = 0 = \min\{d(\underline{z}, \underline{c}) : \underline{z} \in C\}$.
Therefore, **a codeword is always decoded to itself:**

$$\underline{y} \in C \qquad \Longrightarrow \qquad \texttt{DECODE}(\underline{y}) = \underline{y}.$$

How good is a code at correcting errors?

This can be measured in two ways. First, one can determine the maximum number of symbol errors that can occur in a codeword which the decoder is guaranteed to correct. Second, one can calculate the probability $P_{\text{corr}}(C)$ of correct decoding for a specific channel. We will return to these later.

$Rep(3, \mathbb{F}_2)$: an example of a code used for error correction

It is easy to see that the code E_3 is not suitable for error correction. Indeed, suppose the received word was 100. We note that 100 has three nearest neighbours in E_3, namely 000, 110 and 101, all at distance 1. There is no reasonable way to decide which of these was sent, so no efficient decoder.

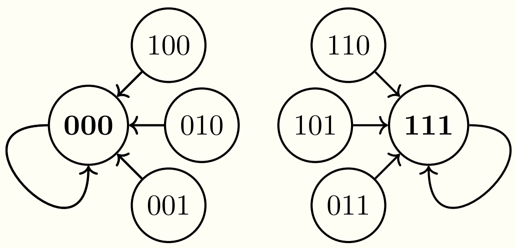
We define a new code, the **binary repetition code of length** 3**.**

Definition: the code $Rep(3, \mathbb{F}_2)$, encoder

$Rep(3, \mathbb{F}_2) = \{000, 111\}$, a subset of $\mathbb{F}_2^3$.
Define $\texttt{ENCODE}\colon \{0,1\} \to Rep(3, \mathbb{F}_2)$ by $\texttt{ENCODE}(0) = 000$, $\texttt{ENCODE}(1) = 111$.

One can observe that every word in $\mathbb{F}_2^3$ has exactly one nearest neighbour in the code $Rep(3, \mathbb{F}_2)$, and here is how the decoder works: for each word $\underline{y} \in \mathbb{F}_2^3$, the arrow points from $\underline{y}$ to $\texttt{DECODE}(\underline{y}) \in Rep(3, \mathbb{F}_2)$.

Example: the decoder for $Rep(3, \mathbb{F}_2)$



Why develop any more theory and not just use E_3 and $Rep(3, \mathbb{F}_2)$?

The problem with these codes is that, for every bit of information, you need to transmit 1.5 bits (using E_3) or 3 bits (using $Rep(3, \mathbb{F}_2)$), because encoding increases the number of bits. Such an increase in transmission costs may be unacceptable, and so more efficient codes need to be designed.

Concluding remarks for Chapter 1

Codes have been used for error correction for thousands of years: a natural language is essentially a code! If we "receive" a corrupted English word such as PHEOEEM, we will assume that is has most likely been THEOREM, because this would involve fewest mistakes.

The following examples are part of historical background to Coding Theory and are not covered in lectures.

Example 1 (a real-world use of Coding Theory in scientific research)

Voyager 1 is a spacecraft launched by NASA in 1977. Its primary mission was to explore Jupiter, Saturn, Uranus and Neptune. Lots of precious photographs and data was sent back

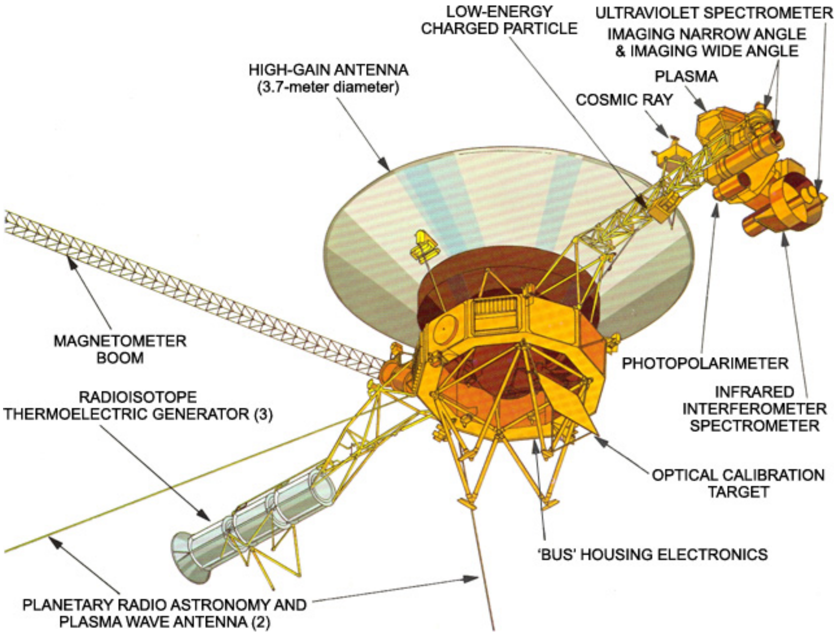


Figure 1.1: The Voyager spacecraft. Image taken from
https://voyager.jpl.nasa.gov/mission/spacecraft/instruments/

to Earth. Later, NASA scientists claimed that Voyager 1 *reached the interstellar space*.

Messages from Voyager 1 travel through the vast expanses of interplanetary space. Given that the spacecraft is equipped with a mere 23 Watt radio transmitter (powered by a plutonium-238 nuclear battery), it is inevitable that noise, such as cosmic rays, interferes with its transmissions. In order to protect the data from distortion, it is encoded with the error-correcting code called *extended binary Golay code*. We will look at this code later in the course. Newer space missions employ more efficient and more sophisticated codes.

Example 2 (CD, a compact disc)

A more down-to-earth example of the use of error-correcting codes. A CD can hold up to 80 minutes of music, represented by an array of zeros and ones. The data on the CD is encoded using a *Reed-Solomon code*. This way, even if a small scratch, a particle of dust or a fingerprint happens to be on the surface of the CD, it will still play perfectly well — all due to error correction.

However, every method has its limits, and larger scratches or stains may lead to something like a thunderclap during playback!

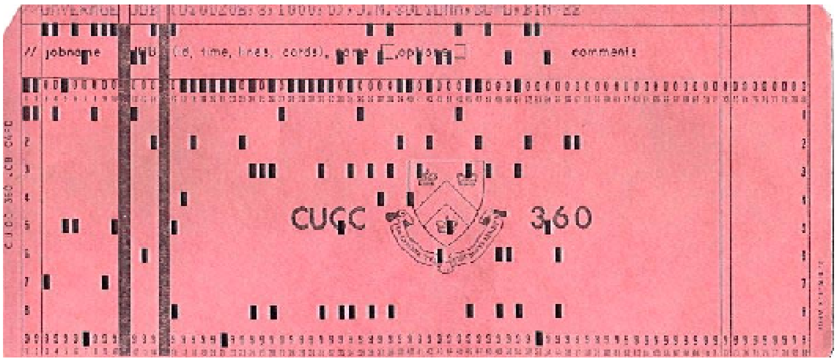


Figure 1.2: A punch card. Image from http://www.columbia.edu/cu/computinghistory

Example 3 (one of the first uses of a code for error correction)

In 1948, Richard Hamming was working at the famous *Bell Laboratories*. Back then, the data for "computers" was stored on *punch cards*: pieces of thick paper where holes represented ones and absences of holes represented zeros. Punchers who had to perforate punch cards sometimes made mistakes, which frustrated Hamming.

Hamming was able to come up with a code with the following properties: each codeword is 7 bits long, and if one error is made in a codeword (i.e., one bit is changed from 0 to 1 or vice versa), one can still recover the original codeword. This made the punch card technology more robust, as a punch card with a few mistakes would still be usable. The trade-off, however, was that the length of data was increased by 75%: there are only 16 different codewords, therefore, they can be used to convey messages which have the length of 4 bits.

The original *Hamming code* will be introduced in the course soon!

Week 1

Exercises (answers at end)

Version 2023-10-01. To accessible online version of these exercises

Exercise 1.1. The **Manchester code** was first used in the Manchester Mark 1 computer at the University of Manchester in 1949 and is still used in low-speed data transfer: e.g. TV remote sending signals via infrared. This binary code consists of two codewords: 10 and 01. The codeword 10 is interpreted by the recipient as the message 0, and 01 is understood to mean 1; whereas the received word 00 or 11 indicates a detected error.

The following error-free fragment of a bit stream encoded by Manchester code had been intercepted: $\ldots 010101x01011010\ldots$ What was the bit x?

Exercise 1.2. Consider the alphabet $\mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The **Luhn checksum** of a word $x_1x_2\ldots x_{16} \in (\mathbb{Z}_{10})^{16}$ is $\pi(x_1) + x_2 + \pi(x_3) + x_4 + \pi(x_5) + \cdots + x_{16}$ mod 10, viewed as an element of $\mathbb{Z}_{10}$. Here $\pi\colon \mathbb{Z}_{10} \to \mathbb{Z}_{10}$ is defined by the rule "$\pi(a)$ is the sum of digits of $2a$". The **Luhn code** consists of all words in $(\mathbb{Z}_{10})^{16}$ whose Luhn checksum is 0.

(i) Write down all values of π and check that π is a permutation of the alphabet $\mathbb{Z}_{10}$.

(ii) Find the total number of codewords of the Luhn code.

(iii) Prove that a single digit error is detected by the Luhn code.

(iv) Look at your 16-digit debit/credit card numbers. Are they codewords of the Luhn code? If you have a card with a number which is **not** a codeword of the Luhn code, can you bring it to the tutorial? Thanks!

Week 1

Exercises — solutions

Version 2023-10-01. To accessible online version of these exercises

Exercise 1.1. The **Manchester code** was first used in the Manchester Mark 1 computer at the University of Manchester in 1949 and is still used in low-speed data transfer: e.g. TV remote sending signals via infrared. This binary code consists of two codewords: 10 and 01. The codeword 10 is interpreted by the recipient as the message 0, and 01 is understood to mean 1; whereas the received word 00 or 11 indicates a detected error.

The following error-free fragment of a bit stream encoded by Manchester code had been intercepted: $\ldots 010101x01011010\ldots$ What was the bit x?

Answer to E1.1. In $\ldots 010101x01011010\ldots$, notice that 11 cannot be a codeword. Therefore, the bit stream is split into codewords in the following way:

$$\ldots 0|10|10|1x|01|01|10|10|\ldots.$$

The codeword $1x$ must be 10 so $x = 0$.

Exercise 1.2. Consider the alphabet $\mathbb{Z}_{10} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$. The **Luhn checksum** of a word $x_1x_2\ldots x_{16} \in (\mathbb{Z}_{10})^{16}$ is $\pi(x_1) + x_2 + \pi(x_3) + x_4 + \pi(x_5) + \cdots + x_{16}$ mod 10, viewed as an element of $\mathbb{Z}_{10}$. Here $\pi\colon \mathbb{Z}_{10} \to \mathbb{Z}_{10}$ is defined by the rule "$\pi(a)$ is the sum of digits of $2a$". The **Luhn code** consists of all words in $(\mathbb{Z}_{10})^{16}$ whose Luhn checksum is 0.

(i) Write down all values of π and check that π is a permutation of the alphabet $\mathbb{Z}_{10}$.

(ii) Find the total number of codewords of the Luhn code.

(iii) Prove that a single digit error is detected by the Luhn code.

(iv) Look at your 16-digit debit/credit card numbers. Are they codewords of the Luhn code? If you have a card with a number which is **not** a codeword of the Luhn code, can you bring it to the tutorial? Thanks!

Answer to E1.2. (i) π is the following permutation of $\mathbb{Z}_{10}$:

$$\begin{pmatrix} 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ 0 & 2 & 4 & 6 & 8 & 1 & 3 & 5 & 7 & 9 \end{pmatrix}.$$

(ii) Every sequence of 15 digits is the beginning of exactly one Luhn codeword. Indeed, let $x_1, \ldots, x_{15} \in \mathbb{Z}_{10}$ be arbitrary. Calculate $z = \pi(x_1)+x_2+\pi(x_3)+x_4+\pi(x_5)+\cdots+\pi(x_{15})$. Then the one and only Luhn codeword of the form $x_1 x_2 \ldots x_{15} x_{16}$ is determined by $z+x_{16} \equiv 0$ mod 10. This is the same as $x_{16} \equiv (-z)$ mod 10.

Therefore, the number of Luhn codewords is equal to the number of sequences of 15 digits, that is, 10^{15}.

(iii) If x_i is replaced by y_i, then the Luhn checksum changes by $y_i - x_i$ mod 10 (if i is even) or by $\pi(y_i) - \pi(x_i)$ mod 10 (if i is odd). In any case, if $y_i \neq x_i$, then neither of these changes is zero mod 10, hence *altering a single digit changes the Luhn checksum*.

A codeword has Luhn checksum 0.

Hence changing a single digit in a codeword gives a word with non-zero Luhn checksum, i.e., not a codeword, resulting in a detected error.

Week 2

Parameters. Bounds

Version 2023-10-01. To accessible online version of this chapter

Synopsis. *Basic properties of a code C can be expressed by numbers called* **parameters.** *We learn why such parameters as the* **rate**, *R, and the* **minimum distance**, *$d(C)$, are important when C is used for channel coding. We also learn to use the notation $(n, M, d)_q$ and $[n, k, d]_q$. It turns out that there is a trade-off between the rate and the minimum distance: both cannot be high (good) at the same time. This trade-off is expressed by inequalities known as* **bounds.** *We only prove the Hamming bound and the Singleton bound in this course, although other bounds have been obtained in coding theory research.*

Parameters of a code

Parameters are numerical characteristics of a code. The most important parameters are:

Definition: parameters of a code

Let F be an alphabet and $C \subseteq F^n$ be a code. Then:

- q denotes the **size of the alphabet,** i.e., $q = \#F$;
- n is called the **length** of the code — each codeword consists of n symbols;
- M denotes the **number of codewords** in the code, i.e., $M = \#C$;
- $k = \log_q M$ is the **information dimension** of C;
- $d(C) = \min\{d(\underline{v}, \underline{w}) : \underline{v}, \underline{w} \in C, \underline{v} \neq \underline{w}\}$ is the **minimum distance** of C;
- $R = k/n$ is the **rate** of C;
- $\delta = d/n$ is the **relative distance** of C.

We say that C is **an** $(n, M, d)_q$**-code** or **an** $[n, k, d]_q$**-code**.

The importance of the minimum distance for error detection and correction

We will now see that the higher $d(C)$, the more symbol errors per codeword is the code C guaranteed to detect and correct.

Notation: $[a]$ denotes the integer part of a real a; e.g., $[3] = [3.5] = [\pi] = 3$, $[7.99] = 7$.

Theorem 2.1: the number of errors detected/corrected by a code

Let C be a code with $d(C) = d$. Throughout the course, t will denote $[(d-1)/2]$. Let $\underline{v} \in C$ and $\underline{y} \in F^n$.

1. If $1 \leq d(\underline{v}, \underline{y}) \leq d - 1$, then $\underline{y} \notin C$. Thus, if at most $d-1$ errors occur in a transmitted codeword, they will be *detected*.
2. If $d(\underline{v}, \underline{y}) \leq t$, then $\underline{y}$ has a unique nearest neighbour in C, which is $\underline{v}$. So if at most t errors occur in a codeword, a decoder will *correct* them by decoding $\underline{y}$ back to $\underline{c}$.

Proof. 1. If $\underline{y} \in C$ then by definition of minimum distance, either $d(\underline{v}, \underline{y}) = 0$ or $d(\underline{v}, \underline{y}) \geq d$. So the statement follows by contrapositive.

2. We use proof by contradiction, so we must assume for contradiction that $\underline{w}$ is a nearest neighbour of $\underline{y}$ in C such that $\underline{w} \neq \underline{v}$. Then $d(\underline{y}, \underline{w}) \leq d(\underline{y}, \underline{v}) \leq t$ so by the triangle inequality

$$0 < d(\underline{v}, \underline{w}) \leq d(\underline{v}, \underline{y}) + d(\underline{y}, \underline{w}) \leq t + t = 2t \leq d - 1.$$

Hence $\underline{v}, \underline{w}$ are distinct codewords at distance less than d. This contradicts d being the minimum distance of C. □

Remark

The Theorem is expressed by saying that a code of minimal distance d **detects up to** $d-1$ **errors** and **corrects up to** $[(d-1)/2]$ **errors** in a codeword.

Channel coding. The importance of rate

To understand why the information dimension k and hence the rate R are defined via logarithm, we recall **channel coding**, the most common use case for codes discussed in the previous chapter. Here is a diagram which shows channel coding with error correction:

$$\boxed{\text{sender}} \xrightarrow[\text{message}]{m} \xrightarrow[\text{codeword}]{\mathtt{ENCODE}(m)} \boxed{\text{channel}} \xrightarrow[\text{word}]{\text{received}} \xrightarrow{\mathtt{DECODE}()} \xrightarrow{\mathtt{ENCODE}^{-1}()} \boxed{\text{receiver}}$$

Messages are arbitrary words of length k, that is, elements of F^k. The cardinality of F^k is q^k, because a word $(u_1, u_2, \ldots, u_k) \in F^k$ can be chosen in q^k ways: q choices for u_1, q independent choices for u_2 and so on. Therefore,

$$M = \#C = \#(F^k) = q^k \qquad \Longrightarrow \qquad k = \log_q M.$$

For each k symbols of information, the sender will transmit a codeword of n symbols. Recall that the rate is $R = \frac{k}{n}$. One has $R \leq 1$ (see the trivial bound below):

Remark: high R, close to 1, is good

Encoding increases transmission costs by a factor of R^{-1}. The higher the rate R, the more economical and efficient the code is.

Although the increase in transmission costs is a proce to pay for error detection or correction, we want this increase to be as small as possible. One can try to construct codes with higher rate, without degrading the error detection or correction performance, by using more sophisticated mathematics. This is one of the main themes in Coding Theory.

There are obstacles to increasing the rate. Mathematically, they are expressed by **bounds.**

Bounds

Proposition 2.2: the trivial bound

If $[n, k, d]_q$-codes exist, then $k \leq n$ and $d \leq n$.

Proof. Let C be an $[n, k, d]_q$-code. Then, by definition, C is a non-empty subset of F^n with $\#F = q$. The cardinality of a set is greater than or equal to the cardinality of its subset. In particular, $M = \#C \leq \#F^n = q^n$. Applying the monotone function $\log_q$ to both sides of the inequality, we obtain $k = \log_q M \leq n$.

Furthermore, the Hamming distance between any two words of length n is an integer between 0 and n. Therefore, $0 < d(C) \leq n$ for any code of length n. □

It is easy to describe the codes which attain $k = n$. All of them are given in the following

Example: F^n, the trivial code of length n

The **trivial code** of length n over the alphabet F is the code $C = F^n$.

Exercise: prove that a code has $k = n$ **if and only if** it is a trivial code. Show that trivial codes have $d = 1$. Show that some codes are not trivial but still have $d = 1$.

We will now give a simple example of codes which attain $d = n$.

Example: $Rep(n, F)$, the repetition code of length n

$Rep(n, F) = \{aaa \dots a \mid a \in F\} \subset F^n$ is the **repetition code** of length n over the alphabet F. All codewords are formed by repeating a symbol n times.

Exercise: prove that $Rep(n, F)$ has $d = n$. Show that some codes are not repetition codes but still have $d = n$.

The Hamming bound

To state the next bound, we recall that $\binom{n}{i}$ is the number of ways to choose i positions out of n. This integer is called the binomial coefficient. It is given by the formula $\binom{n}{i} = \dfrac{n!}{(n-i)!\,i!} = \dfrac{n(n-1)\dots(n-i+1)}{1 \cdot 2 \cdot \dots \cdot i}$.

Theorem 2.3: the Hamming bound

Denote $t = [(d-1)/2]$. If $(n, M, d)_q$-codes exist, $M \leq \dfrac{q^n}{\sum\limits_{i=0}^{t} \binom{n}{i}(q-1)^i}$.

Before proving the Theorem, we introduce

Definition: Hamming sphere

If $\underline{y} \in F^n$ and $r \leq n$, the **Hamming sphere** with centre $\underline{y}$ and radius r is the set

$$S_r(\underline{y}) = \{\underline{v} \in F^n : d(\underline{v}, \underline{y}) \leq r\}.$$

The number of words in the Hamming sphere depends only on the radius r (not on $\underline{y}$):

Lemma 2.4: the cardinality of a Hamming sphere

$$\#S_r(\underline{y}) = \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

Proof. To construct a word $\underline{v}$ at distance i from $\underline{y}$, we need to choose i positions out of n where $\underline{y}$ will differ from $\underline{v}$. Then we need to change the symbol in each of the i chosen positions to one of the other $q-1$ symbols. The total number of choices for $\underline{v}$ which is at distance exactly i from $\underline{y}$ is thus $\binom{n}{i}(q-1)^i$.

The Hamming sphere contains all vectors at distance $0 \le i \le r$ from $\underline{v}$, so we sum over i from 0 up to r. The Lemma is proved. $\square$

Proof of Theorem 2.3. First of all, we prove that spheres of radius t centred at distinct codewords $\underline{c}$ do not overlap. Indeed, by Theorem 2.1(2), each word in $S_t(\underline{c})$ has unique nearest neighbour, which is $\underline{c}$. Hence a word in $S_t(\underline{c})$ cannot lie in another such sphere (a word cannot have two *unique* nearest neighbours!)

Hence the whole set F^n contains M *disjoint* spheres centred at codewords. By Lemma 2.4, each of the M spheres contains $\sum_{i=0}^{t}\binom{n}{i}(q-1)^i$ words. The number of elements in a disjoint union of sets is equal to the sum of cardinalities of the sets, hence the total number of words in the M spheres is $M\sum_{i=0}^{t}\binom{n}{i}(q-1)^i$. Since the union of the M spheres is a subset of F^n, this does not exceed $\#F^n = q^n$. The bound follows. $\square$

Given the length n and the minimum distance d, we may wish to know whether there are codes with the number of codewords *equal* to the Hamming bound. Such a code would be the most economical (highest possible number M of codewords). Such codes have a special name:

Definition: perfect code

A code which attains the Hamming bound is called a **perfect** code.

It turns out that meaningful perfect codes are quite rare. When the number of symbols in the alphabet is a prime power, a complete classification of perfect codes up to parameter equivalence is known; we will see it later in the course.

Remark: what does it mean to attain the bound?

Attains the bound means: the inequality in the bound becomes equality for this code. It is a mistake to say that perfect codes are those that "satisfy" the Hamming bound. Every code *satisfies* the Hamming bound — only perfect codes *attain* it!

The Singleton bound

Another upper bound on the number M of codewords can be conveniently stated for $k = \log_q M$.

Theorem 2.5: the Singleton bound

If $[n,k,d]_q$ codes exist, $k \le n-d+1$.

Proof. Let C be an $[n,k,d]_q$-code. Consider the function $f\colon C \to F^{n-d+1}$ where $f(\underline{v})$ is the word obtained from $\underline{v}$ by deleting the last $d-1$ symbols.

I claim that f is an injective function. Indeed, if $\underline{v},\underline{w} \in C$, $\underline{v} \neq \underline{w}$, then by definition of the minimum distance, $\underline{v}$ and $\underline{w}$ differ in at least d positions. Since f deletes only $d-1$ symbols, the words $f(\underline{v})$ and $f(\underline{w})$ still differ in at least one position. So $f(\underline{v}) \neq f(\underline{w})$. Injectivity of f is proved.

Now, by the Pigeonhole Principle, injective functions $f\colon C \to F^{n-d+1}$ exist only if $\#C \le \#F^{n-d+1}$. We conclude that $\#C \le q^{n-d+1}$ so that $k = \log_q \#C \le n-d+1$ as claimed. $\square$

Definition: maximum distance separable code, MDS code

A code which attains the Singleton bound is called a **maximum distance separable (MDS)** code.

Remark: the bounds do not work in reverse

It is important to remember that the converses to Theorems 2.3 and 2.5 do not hold. That is, if the numbers n,k,d,q satisfy the Hamming bound and the Singleton bound, it **does not imply that an $[n,k,d]_q$-code exists.** For example, n,k,d,q may fail further bounds, not covered in this course.

Week 2

Exercises (answers at end)

Version 2023-10-12. To accessible online version of these exercises

Exercise 2.1. Consider the Manchester code and the Luhn code. For each of these codes, determine the parameters $[n, k, d]_q$ of the code; state how many errors the code can detect and how many errors the code can correct; determine if the code is perfect and/or MDS.

Exercise 2.2. [*alternative way to check that a code is perfect; may need this for the exam*] The proof of the Hamming bound in the lecture notes shows that a code $C \subseteq F^n$ is perfect, if and only if the (disjoint) spheres of radius $t = [(d(C) - 1)/2]$, centred at codewords of C, fully cover the set F^n of all words.

Equivalently, C is perfect iff every word in F^n is at distance $\leq t$ from some codeword.

(a) Prove that a perfect code has odd minimum distance d.
(*Hint*: if d is even, construct a word at distance $d/2$ from a codeword and show that it is not at distance $\leq t$ from any codeword.)

(b) Show that binary repetition codes of odd lengths are perfect.

Exercise 2.3 (not done in the tutorial)**.** Show that Rep(n, F) is not perfect if $q = \#F > 2$. (*Hint*: using three different symbols, write down a word at distance $> n/2$ from each codeword.)

Exercise 2.4 (not done in the tutorial)**.** Assume that the cost of transmitting one symbol via a q-ary channel is cq. (*Imagine a q-ary channel as a cable with q wires; the costs of building and maintaining it would be roughly proportional to q.*) Suppose that you are given a very large number M and need to design a code with M codewords. You have the control over the length n and the size q of the alphabet. Which q will ensure the lowest transmission costs *per codeword*? In particular, are the binary channels (the type most widely used in today's computer networks) the most economical?

Week 2

Exercises — solutions

Version 2023-10-12. To accessible online version of these exercises

Exercise 2.1. Consider the Manchester code and the Luhn code. For each of these codes, determine the parameters $[n, k, d]_q$ of the code; state how many errors the code can detect and how many errors the code can correct; determine if the code is perfect and/or MDS.

Answer to E2.1.

The Manchester code: $n = 2$, $q = 2$, $M = 2$ so $k = 1$; $d = 2$ by inspection. A $[2, 1, 2]_2$-code, can detect up to 1 bit error. Does not correct errors. Is not perfect as $M = 2 < 2^2 / \sum_{i=0}^{0} \binom{2}{0}(2-1)^0$ (using $t = 0$). Is MDS as $1 = 2 - 2 + 1$.

The Luhn code: $n = 16$, $q = 10$, $M = 10^{15}$ so $k = 15$. One has $d \leq 2$ as for example the codewords $0000\ldots0$ and $9100\ldots0$ are at distance 2.

On the other hand, $d > 1$. Indeed, we showed last week that changing exactly one symbol in a Luhn codeword cannot give a Luhn codeword. Hence no two codewords are at distance 1.

To conclude, the Luhn code is a $[16, 15, 2]_{10}$-code.

Can detect up to 1 symbol error. Does not correct errors.

One has $n - d + 1 = 16 - 2 + 1 = 15 = k$ — this is an MDS code. (It is not perfect; this can be checked directly, or use an exercise below as d is even.)

Exercise 2.2. [*alternative way to check that a code is perfect; may need this for the exam*] The proof of the Hamming bound in the lecture notes shows that a code $C \subseteq F^n$ is perfect, if and only if the (disjoint) spheres of radius $t = [(d(C) - 1)/2]$, centred at codewords of C, fully cover the set F^n of all words.

Equivalently, C is perfect iff every word in F^n is at distance $\leq t$ from some codeword.

(a) Prove that a perfect code has odd minimum distance d.
(*Hint*: if d is even, construct a word at distance $d/2$ from a codeword and show that it is not at distance $\leq t$ from any codeword.)

(b) Show that binary repetition codes of odd lengths are perfect.

Answer to E2.2. First of all, we explain why the equivalent way to check that the code is perfect is valid. We know from the proof of Theorem 2.3 that the $\#C$ spheres $S_t(\underline{c})$, where $\underline{c} \in C$, are disjoint. Each sphere contains $\sum_{i=0}^{t} \binom{n}{i}(q-1)^i$ words, hence the total number of words covered by these spheres is $(\#C)\sum_{i=0}^{t} \binom{n}{i}(q-1)^i$. This number is equal to $\#F^n = q^n$ iff these spheres cover all words in F^n. On the other hand, this number is equal to $\#F^n = q^n$ iff the code C is perfect. Q.E.D.

(a) Assume for contradiction that a perfect C has even $d(C) = d$. Take any codeword $\underline{w}$ of C and change the first $d/2$ symbols in $\underline{w}$ to obtain a word $\underline{z} \in F^n$ with $d(\underline{z}, \underline{w}) = d/2$. Since C is perfect, there must be another codeword $\underline{v}$ such that $d(\underline{v}, \underline{z}) \leq t$. Then by the triangle inequality $d(\underline{v}, \underline{w}) \leq t + d/2 \leq (d-1)/2 + d/2 < d$, a contradiction.

(b) Rep$(n, \mathbb{F}_2)$ consists of $\underline{0} = 00\ldots0$ and $\underline{1} = 11\ldots1$. Here $n = 2t+1$. A word $\underline{y} \in \mathbb{F}_2^n$ may have $\leq t$ zero bits — then $d(\underline{y}, \underline{1}) \leq t$. Otherwise, $\underline{y}$ has $t+1$ or more zero bits, hence $\leq t$ one bits, and $d(\underline{y}, \underline{0}) \leq t$. This shows that every word is at distance $\leq t$ from one of the two codewords.

Exercise 2.3 (not done in the tutorial)**.** Show that Rep(n, F) is not perfect if $q = \#F > 2$. (*Hint*: using three different symbols, write down a word at distance $> n/2$ from each codeword.)

Answer to E2.3. Assume that the alphabet F contains at least three symbols; for simplicity, let F contain 0, 1 and 2. The repetition code Rep(n, F) has minimum distance n, hence $t = [(n-1)/2]$.

Let n be odd — the case of even n follows from 2.2(a). Then $n = 2t+1$. Consider the word $0\ldots01\ldots12$ which has t zeros, t ones and 1 two. It differs from each codeword in at least $t+1$ positions, hence is not at distance $\leq t$ from any codeword. We have shown that the code is not perfect.

Exercise 2.4 (not done in the tutorial)**.** Assume that the cost of transmitting one symbol via a q-ary channel is cq. (*Imagine a q-ary channel as a cable with q wires; the costs of building and maintaining it would be roughly proportional to q.*) Suppose that you are given a very large number M and need to design a code with M codewords. You have the control over the length n and the size q of the alphabet. Which q will ensure the lowest transmission costs *per codeword*? In particular, are the binary channels (the type most widely used in today's computer networks) the most economical?

Answer to E2.4. The cost of transmitting one codeword is cqn. By the trivial bound, $n \geq k = \log_q M$ so this cost is estimated from below as $cq \log_q M = K \dfrac{q}{\ln q}$ where the constant K is $c \ln M$. The function $f(x) = x/\ln x$ decreases on $(0, e)$ and increases on (e, ∞) (*check this by differentiation or otherwise*) so $f(q) > f(3)$ if $q > 3$. Hence the only candidates for the minimum are $q = 2$ and $q = 3$. Calculating $f(2) \cong 2.89$ and $f(3) \cong 2.73$, we conclude that — if we accept the (somewhat arbitrary) assumptions in the problem — ternary codes are the most economical.

Week 3

Linear codes

Version 2023-10-02. To accessible online version of this chapter

Synopsis. *The most important class of codes is linear codes. Their ability to correct errors is no worse than that of general codes, but linear codes are easier to implement in practice and allow us to use algebraic methods. We learn how to find the minimum distance by looking at weights, and how to define a linear code by its generator matrix.*

The definition of a linear code

Reminder (vector spaces): let $\mathbb{F}_q$ denote the field of q elements. When we use $\mathbb{F}_q$ as the alphabet, we refer to words in $\mathbb{F}_q^n$ as (row) **vectors**. The set $\mathbb{F}_q^n$ of all vectors of length n has the structure of a **vector space** over the field $\mathbb{F}_q$. If the vectors $\underline{u}$, $\underline{v}$ are in $\mathbb{F}_q^n$, we can add the vectors together: $\underline{u} + \underline{v} \in \mathbb{F}_q^n$, and multiply a vector by a scalar: $\lambda\underline{u} \in \mathbb{F}_q^n$ for all $\lambda \in \mathbb{F}_q$. The addition and the scalar multiplication are performed **componentwise**. We will often write vectors in compact form, as words:

$$011011, 100110 \in \mathbb{F}_2^6 \qquad \mapsto \qquad 011011 + 100110 = 111101 \in \mathbb{F}_2^6.$$

Definition: linear code, codevector

A **linear code** is a subspace of the vector space $\mathbb{F}_q^n$.
Codewords of a linear code are called **codevectors.**

This means that the zero vector $\underline{0}$ belongs to C, and that sums and scalar multiples of codevectors are again codevectors. Thus, C is a vector space in its own right.

Discussion: Why are linear codes useful? (not examinable)

1. They seem to be as efficient as general codes. In particular, it was proved that Shannon's Theorem about the capacity of a channel (discussed later) is still true for linear codes.

2. It is possible to define a linear code without specifying all the codewords (see below).

3. The minimum distance is easier to calculate than for general codes (see below).

4. We can use algebra to design linear codes and to construct efficient encoding and decoding algorithms.

The absolute majority of codes designed by coding theorists are linear codes. In the rest of the course, (almost) all the codes we consider will be linear codes.

End of discussion.

Example: trivial, repetition codes

The trivial code $\mathbb{F}_q^n$ is a linear code. (Indeed, $\mathbb{F}_q^n$ is a vector subspace of itself.)
The repetition code $Rep(n, \mathbb{F}_q)$ over $\mathbb{F}_q$ is a linear code (*exercise; will see soon*).

To get non-trivial examples, we need to introduce more structure.

The weight

Definition: weight of a vector, weight of a code

The **weight** $w(\underline{v})$ **of a vector** $\underline{v} \in \mathbb{F}_q^n$ is the number of non-zero symbols in $\underline{v}$.
The **weight** $w(C)$ **of a code** $C \subseteq \mathbb{F}_q^n$ is $w(C) = \min\{w(\underline{v}) \mid \underline{v} \in C \setminus \{\underline{0}\}\}$.

Lemma 3.1: distance and weight

For any vectors $\underline{v}, \underline{y} \in \mathbb{F}_q^n$, $d(\underline{v}, \underline{y}) = w(\underline{v} - \underline{y})$.

Proof. Indeed, $d(\underline{v}, \underline{y})$ is the number of positions i, $1 \le i \le n$, where $v_i \neq y_i$. Obviously, this is the same as the number of positions i where $v_i - y_i \neq 0$. By definition of the weight, this is $w(\underline{v} - \underline{y})$, as claimed. □

Recall that the minimum distance, $d(C)$, of a code C is a very important parameter which tells us how many errors can the code detect and correct in a codeword. The following theorem shows how one can find $d(C)$ if C is linear.

Theorem 3.2: minimum distance equals weight

$d(C) = w(C)$ for a linear code C.

Proof. Take a codevector $\underline{v}$ such that $w(C) = w(\underline{v})$. Observe, $w(\underline{v}) = w(\underline{v} - \underline{0}) = d(\underline{v}, \underline{0})$ but $\underline{v} \neq \underline{0} \in C$ so $w(\underline{v}) \ge d(C)$. We proved that $w(C) \ge d(C)$.

Now take a pair $\underline{y} \neq \underline{z} \in C$ such that $d(\underline{y}, \underline{z}) = d(C)$. Rewrite this as $w(\underline{y} - \underline{z})$. Since C is linear, $\underline{y} - \underline{z} \in C \setminus \{0\}$ so $w(\underline{y} - \underline{z}) \geq w(C)$. We proved that $d(C) \geq w(C)$. $\square$

Remark: in the proof, we twice used that C is linear: first, $\underline{0} \in C$; second, $\underline{y}, \underline{z} \in C$ implies $\underline{y} - \underline{z} \in C$. This condition is essential.

Remark: given a linear code C, one needs to check only $M - 1$ vectors to compute $d(C) = w(C)$. For a non-linear code, one has to check $M(M-1)/2$ pairs of words to compute the minimum distance d.

Here is a non-trivial construction of a linear code.

Example: the zero sum code

For any finite field $\mathbb{F}_q$ and for any $n \geq 1$ we can define the **zero sum code** in $\mathbb{F}_q^n$ as

$$Z = \{(v_1, v_2, \dots, v_n) \in \mathbb{F}_q^n \mid v_1 + v_2 + \dots + v_n = 0 \text{ in } \mathbb{F}_q\}.$$

We note that the zero sum code in $\mathbb{F}_q^n$ is a linear code because Z is the set of solutions to the homogeneous linear equation $v_1 + \dots + v_n = 0$. It is known from linear algebra (and is easy to check directly) that the sum of two vectors satisfying this equation also satisfies this equation, and scaling a vector satisfying this equation again satisfies the equation. In other words, Z is a vector space.

Binary zero sum codes are very common and have a special name.

Example: The binary even weight code E_n

The **binary even weight code of length** n is defined as

$$E_n = \{\underline{v} \in \mathbb{F}_2^n : w(\underline{v}) \text{ is even}\}.$$

Due to the rules of arithmetic in $\mathbb{F}_2$ we have

$$E_n = \{x_1 x_2 \dots x_n : x_i \in \mathbb{F}_2,\ x_1 + x_2 + \dots + x_n = 0 \text{ in } \mathbb{F}_2\}$$

which shows that E_n is a particular case of a zero sum code, hence is a linear code.

Note: 0 is an even number! The binary even weight code contains the codeword $00\dots0$.

Basic properties of the binary even weight code E_n

Minimum distance = weight: a vector with only one 1 has odd weight but a vector $1100\dots0$ of weight 2 is in E_n. Hence $d(E_n) = w(E_n) = 2$. The code detects up to 1 error and corrects up to 0 errors.

The number of codewords: in a codeword $\underline{v} = (x_1, x_2, \dots, x_n)$, the first $n-1$ bits can be arbitrary (2^{n-1} combinations), and the last bit is uniquely determined by $x_n = x_1 + \ldots + x_{n-1}$, where $+$ is the addition is in the field $\mathbb{F}_2$. We thus have 2^{n-1} codewords.

Another argument to that effect is as follows. We can take a binary word and flip (change) its first bit. This operation splits the set $\mathbb{F}_2^n$ into pairs of vectors, such that the vectors in a pair only differ in the first bit. Each pair contains one vector of even weight and one vector of odd weight. Therefore, the number of vectors of even weight is equal to the number of vectors of odd weight, and is $\frac{1}{2}\#\mathbb{F}_2^n = 2^{n-1}$.

Conclusion: E_n is an $[n, n-1, 2]_2$-code.

Remark: A widely used code. If an error is *detected*, the recipient will request retransmission of the codeword where the error occurred. Error *correction* is not available.

The code generated by a matrix. A generator matrix of a linear code

We have an unlimited supply of linear codes, due to the following construction.

Definition: the linear code generated by a matrix

Let G be a $k \times n$ matrix with linearly independent rows $\underline{r}_1, \dots, \underline{r}_k \in \mathbb{F}_q^n$. The code

$$C = \{u_1\underline{r}_1 + \ldots + u_k\underline{r}_k \mid u_1, \dots, u_k \in \mathbb{F}_q\} \quad \subseteq \quad \mathbb{F}_q^n$$

is said to be **generated by the matrix** G. In this case, the function

$$\texttt{ENCODE}\colon \mathbb{F}_q^k \to C, \qquad \texttt{ENCODE}(\underline{u}) = \underline{u}G \quad \text{for all } \underline{u} \in \mathbb{F}_q^k$$

is the **encoder** for C given by the matrix G.

Proposition 3.3: properties of a code generated by a matrix

In the above definition, C is a linear code. The function ENCODE is a bijective linear map between $\mathbb{F}_q^k$ and C. The **information dimension** of C is k and **is equal to vector space dimension,** $\dim C$.

Proof. The definition says that C is the span of $\underline{r}_1, \dots, \underline{r}_k$ in the vector space $\mathbb{F}_q^n$. By linear algebra, a span is a subspace of $\mathbb{F}_q^n$ hence a linear code.

Matrix multiplication is linear in each argument so $\texttt{ENCODE}(\underline{u}) = \underline{u}G$ is a linear function of $\underline{u} = (u_1, \dots, u_k)$. As C consists of vectors of the form $u_1\underline{r}_1 + \cdots + u_k\underline{r}_k = \underline{u}G$, the image of ENCODE is C so ENCODE is surjective. The kernel of ENCODE is made up of all

$(u_1, \ldots, u_k)$ such that $u_1\underline{r}_1 + \ldots + u_k\underline{r}_k = \underline{0}$, but as $\underline{r}_1, \ldots, \underline{r}_k$ are linearly independent, $\ker \texttt{ENCODE} = \{\underline{0}\}$ and so ENCODE is injective, hence bijective.

Hence $M = \#C = \#\mathbb{F}_q^k = q^k$ and so the information dimension of C is $\log_q(M) = k$.

On the other hand, the vector space dimension of C is, by definition, the number of element in a basis of C. Note that the k-element set $\{\underline{r}_1, \ldots, \underline{r}_k\}$ is a basis of C, as this is a linearly independent set which spans C. Hence $\dim C$ is also k. ☐

In fact, **all** linear codes arise from the above construction. Indeed, we know from linear algebra that every vector space C has a basis. So every linear code is generated by a matrix:

Definition: generator matrix

Let $C \subseteq \mathbb{F}_q^n$ be a linear code. A **generator matrix** of C is a matrix $G = \begin{bmatrix} \underline{r}_1 \\ \underline{r}_2 \\ \vdots \\ \underline{r}_k \end{bmatrix}$, where the row vectors $\underline{r}_1, \ldots, \underline{r}_k$ are a basis of C. (Clearly, C is generated by any of its generator matrices.)

Let us consider some simple matrices and work out the codes they generate.

Example: matrices that can generate a trivial code

The identity matrix I_n is a generator matrix for the trivial code, $\mathbb{F}_q^n$. Any other $n \times n$ matrix with linearly independent rows is also a generator matrix for the trivial code of length n.

Example: matrices that generate repetition codes

The repetition code $Rep(n, \mathbb{F}_q)$ has generator matrix $G = \begin{bmatrix} 1 & 1 & \ldots & 1 \end{bmatrix}$, of size $1 \times n$. The matrix λG for any $\lambda \in \mathbb{F}_q$, $\lambda \neq 0$ is also a generator matrix for $Rep(n, \mathbb{F}_q)$.

Example: matrices that generate the binary even weight code E_3

$E_3 = \{000, 011, 101, 110\}$ has $4 = 2^2$ codewords, so the dimension of this code is 2. Therefore, a generator matrix has 2 rows and 3 columns.
To write down a generator matrix, we need to take two linearly independent codevectors. We must not use the zero codevector, 000, because a linearly independent

set must not contain the zero vector, but can use any two others. So, each of

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

is a generator matrix for E_3.

Discussion: storing generator matrix instead of the whole code

Thus, to work with a linear code, it is enough to store just its generator matrix instead of storing all codevectors. This approach to linear codes has its practical advantages and disadvantages.

The single **advantage** which outweighs everything else is the amount of storage space required.

> To visualise the difference between storing all the q^k codewords of a linear code and storing only k rows of a generator matrix, consider a binary code of dimension about 1500 used in computer networking for error detection. We can store 1500 rows of a generator matrix, but it is absolutely impossible to store a list of all 2^{1500} codewords. Indeed, the number 10^{100} (the *googol*) is believed to be bigger than the number of electrons in the visible Universe; but googol is less than 2^{340}.

Disadvantages. A generator matrix is in general **not unique**, because a basis of a vector space C can be chosen in more than one way. It may not be obvious if two matrices generate the same code (although it is easy to test by bringing both matrices to reduced row echelon form and comparing the result).

If a linear code C is specified by a generator matrix G, it may be difficult to compute the **weight** $w(C)$ of C. Of course, the weight of C does not exceed, but is in general not equal to, the minimum weight of a row of G. For some linear codes which have been used in practice, the weight is not known!

Generator matrices in standard form

For a linear code C, the encoder, $\texttt{ENCODE}(\underline{u}) = \underline{u}G$, depends on the choice of a generator matrix G. In practice, for many codes there is the best choice:

Definition: matrix in standard form

A matrix G is in *standard form* if its leftmost colums form an identity matrix:

$$G = [I_k \,|\, A] = \left[\begin{array}{cccc|ccc} 1 & 0 & \dots & 0 & * & \dots & * \\ 0 & 1 & \dots & 0 & * & \dots & * \\ & & \ddots & & & & \\ 0 & 0 & \dots & 1 & * & \dots & * \end{array}\right].$$

Note that entries in the last $n-k$ columns, denoted $*$, are arbitrary elements of $\mathbb{F}_q$.

If G is in standard form, then, after encoding, the first k symbols of the codeword show the original message:

$$\underline{u} \in \mathbb{F}_q^k \quad \mapsto \quad \mathtt{ENCODE}(\underline{u}) = \underline{u}G = \underline{u}[I_k \,|\, A] = [\underline{u} \,|\, \underline{u}A]$$

(this is an easy example of multiplication of block matrices). This means that it is easy to **unencode** a codevector, simply by taking its first k symbols.

In this situation, the first k symbols of a codeword are called *information symbols*. The last $n-k$ symbols are called *check symbols*; their job is to protect the information from noise by increasing the Hamming distance between codewords.

Theorem 3.4: generator matrix in standard form

If a generator matrix in standard form exists for a linear code C, it is unique, and any generator matrix can be brought to the standard from by the following operations:

(R1) Permutation of rows.

(R2) Multiplication of a row by a non-zero scalar.

(R3) Adding a scalar multiple of one row to another row.

Proof. Not given — a standard fact from linear algebra (uniqueness of reduced row echelon form). We will do examples to show how to find the generator matrix in standard form. □

Remark. If we apply a sequence of the row operations (R1), (R2) and (R3) to a generator matrix of a code C, we again obtain a generator matrix of C. This is implied in the Theorem, and follows from the fact that a basis of a vector space remains a basis under permutations, multiplication of an element of the basis by a scalar, and adding a scalar multiple of an element to another element. This fact is known from linear algebra.

Examples of finding a generator matrix in standard form, and some codes which have no generator matrix in standard form, are on example sheets. We consider one example here:

Example: bringing a generator matrix into standard form

The binary code C is generated by $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$. Find the generator matrix in standard form for C. Find the parameters of C. Identify the code C by its well-known name.

Solution: apply row operations $\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$ $(r_1 \leftrightarrow r_2)$ $\begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{bmatrix}$ $(r_3 \to r_3 + r_1, r_4 \to r_4 + r_1)$ $\begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$ $(r_2 \leftrightarrow r_4)$ $\begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ $(r_3 \to r_3 + r_2, r_4 \to r_4 + r_2)$ $\begin{bmatrix} 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$ $(r_1 \to r_1 + r_4)$ $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$ $(r_4 \to r_4 + r_3)$ $\begin{bmatrix} 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$.

The parameters of C are: length 5 (the number of columns of the generator matrix), dimension 4 (the number of rows of the generator matrix). From the generator matrix in standard form (its rows are also codevectors!) we can see that $w(C) \le 2$. In fact, all the rows of the generator matrix are of even weight; hence they lie in the vector space E_5. Hence all their linear combinations lie in E_5. Since $\dim C = 4 = \dim E_5$, we have $C = E_5$ (*the even weight code of length 5*) and $d(C) = w(C) = 2$.

Week 3

Exercises (answers at end)

Version 2023-10-02. To accessible online version of these exercises

Exercise 3.1. Write down a generator matrix for the repetition code $Rep(5, \mathbb{F}_7)$.

Exercise 3.2 (important — you need to know the ISBN-10 code for the exam). Consider the field $\mathbb{F}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X\}$ of integers modulo 11; by convention, X means ten. The **ISBN-10 checksum** of a word $x_1 x_2 \ldots x_{10}$ in $\mathbb{F}_{11}^{10}$ is

$$1x_1 + 2x_2 + \cdots + 10x_{10} = \sum_{i=1}^{10} i x_i \in \mathbb{F}_{11}.$$

The **ISBN-10** code, which was used to give unique IDs to books until it was superseded by ISBN-13, consists of all vectors in $\mathbb{F}_{11}^{10}$ which have zero checksum. It is a linear code (the set of solutions to a homogeneous linear equation is a vector space).

Show that $d = 2$ hence the code is not perfect. Deduce that the code detects a single error.

Exercise 3.3 (an exam style question)**.** Let C be the ternary linear code generated by $G = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 1 & 1 \end{bmatrix}$. (Reminder: *ternary* means that the alphabet is $\mathbb{F}_3$.)

(a) List all the codevectors of C. Find $d(C)$ by inspection. Deduce that C is a perfect code. Does C attain the Singleton bound?

(b) Find a generator matrix of C in standard form.

Exercise 3.4 (not done in tutorial)**.** Show that the ISBN-10 code detects a transposition error (when two unequal adjacent digits are swapped in a codeword, it is no longer a codeword).

Exercise 3.5 (not done in tutorial)**.** (a) Show that the binary linear code generated by $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ has no generator matrix in standard form.

(b) The ISBN-10 code has a generator matrix in standard form. Find it.

Week 3

Exercises — solutions

Version 2023-10-02. To accessible online version of these exercises

Exercise 3.1. Write down a generator matrix for the repetition code $Rep(5, \mathbb{F}_7)$.

Answer to E3.1. $Rep(n, \mathbb{F}_q)$ consists of all vectors proportional to the vector $11\ldots11$ of n ones. Hence the only row of the $1 \times n$ matrix $\begin{bmatrix}1 & 1 & \ldots & 1 & 1\end{bmatrix}$ spans the code, i.e., forms a spanning set, which is obviously linearly independent.

Multiplying the above generator matrix by any scalar $\lambda \in \mathbb{F}_q \setminus \{0\}$ also gives a generator matrix for $Rep(n, \mathbb{F}_q)$.

For $Rep(5, \mathbb{F}_7)$ we get $\begin{bmatrix}1 & 1 & 1 & 1 & 1\end{bmatrix}$ or any matrix obtained by scaling this one by a non-zero scalar in $\mathbb{F}_7$.

Exercise 3.2 (important — you need to know the ISBN-10 code for the exam). Consider the field $\mathbb{F}_{11} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, X\}$ of integers modulo 11; by convention, X means ten. The **ISBN-10 checksum** of a word $x_1x_2\ldots x_{10}$ in $\mathbb{F}_{11}^{10}$ is

$$1x_1 + 2x_2 + \cdots + 10x_{10} = \sum_{i=1}^{10} ix_i \in \mathbb{F}_{11}.$$

The **ISBN-10** code, which was used to give unique IDs to books until it was superseded by ISBN-13, consists of all vectors in $\mathbb{F}_{11}^{10}$ which have zero checksum. It is a linear code (the set of solutions to a homogeneous linear equation is a vector space).

Show that $d = 2$ hence the code is not perfect. Deduce that the code detects a single error.

Answer to E3.2.

To show that $w(C) \leq 2$, take the codevector $\underline{v} = 0000110000$. We have $w(C) \leq w(\underline{v}) = 2$.

On the other hand, a vector $00\dots0x_i0\dots0$ of weight 1 cannot be in the code: its checksum is ix_i mod 11. Neither i nor x_i is zero mod 11, and 11 is a prime, so ix_i is not zero mod 11. Thus, $w(C) > 1$.

It follows that $w(C) = 2$. Since $d(C) = w(C)$ for linear codes, we have $d(C) = 2$.

Exercise 3.3 (an exam style question)**.** Let C be the ternary linear code generated by $G = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 1 & 1 \end{bmatrix}$. (Reminder: *ternary* means that the alphabet is $\mathbb{F}_3$.)

(a) List all the codevectors of C. Find $d(C)$ by inspection. Deduce that C is a perfect code. Does C attain the Singleton bound?

(b) Find a generator matrix of C in standard form.

Answer to E3.3. (a) First of all, it is useful to recall that the total number of codevectors is q^k where k is the number of rows in the generator matrix. In this case, $q^k = 3^2 = 9$.

We need to list all the 9 linear combinations of the two rows of the matrix G,

$$\underline{r}_1 = [0 \quad 1 \quad 2 \quad 1], \quad \underline{r}_2 = [2 \quad 0 \quad 1 \quad 1].$$

Let us do this by encoding all vectors of length 2, in matrix form. Remember, 'encoding in matrix form' simply means that the codevector $\lambda_1\underline{r}_1 + \lambda_2\underline{r}_2$ is written as $[\lambda_1 \quad \lambda_2]G$ where $G = \begin{bmatrix} \underline{r}_1 \\ \underline{r}_2 \end{bmatrix}$:

$[0 \quad 0]G = [0 \quad 0 \quad 0 \quad 0]$, $[0 \quad 1]G = [2 \quad 0 \quad 1 \quad 1]$, $[0 \quad 2]G = [1 \quad 0 \quad 2 \quad 2]$,
$[1 \quad 0]G = [0 \quad 1 \quad 2 \quad 1]$, $[1 \quad 1]G = [2 \quad 1 \quad 0 \quad 2]$, $[1 \quad 2]G = [1 \quad 1 \quad 1 \quad 0]$,
$[2 \quad 0]G = [0 \quad 2 \quad 1 \quad 2]$, $[2 \quad 1]G = [2 \quad 2 \quad 2 \quad 0]$, $[2 \quad 2]G = [1 \quad 2 \quad 0 \quad 1]$.

To find $d(C)$, one could of course check all 36 pairwise distances between codewords — but this is wrong, because we know that C is a linear code, so $d(C) = w(C)$. We check each of the 8 non-zero codevectors obtained above and conclude that $w(C) = 3$.

To show that C is perfect, let us check the Hamming bound (in logarithmic form): $t = 1$ so $k = n - \log_q(\binom{n}{0} + \binom{n}{1}(q-1))$, $2 = 4 - \log_3(1 + 4 \times 2)$, $2 = 4 - \log_3 9$ — true. Hence the code is perfect.

Exercise: show that C is an MDS code (attains the Singleton bound).

(b) Once **we know all the codevectors**, a generator matrix in standard form does not require any further calculations. Simply select the codevectors which begin with 10 and 01:

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

Exercise 3.4 (not done in tutorial)**.** Show that the ISBN-10 code detects a transposition error (when two unequal adjacent digits are swapped in a codeword, it is no longer a codeword).

Answer to E3.4. We will show that when two adjacent unequal symbols in a vector are swapped, the checksum of the vector is changed. Indeed, suppose a vector $\ldots xy \ldots$ (symbols in positions i, $i+1$) is changed to $\ldots yx \ldots$. The checksum changes by $iy + (i+1)x - (ix + (i+1)y) = x - y$. This is not zero mod 11 as long as $x \neq y$.

Therefore, a codeword (with checksum zero) becomes a non-codeword (with checksum not zero) as a result of a transposition error.

Exercise 3.5 (not done in tutorial)**.** (a) Show that the binary linear code generated by $\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$ has no generator matrix in standard form.

(b) The ISBN-10 code has a generator matrix in standard form. Find it.

Answer to E3.5. (a) The codevectors of this code are 0000, 1100, 0011 and 1111. The first row of a generator matrix in standard form must be a codevector and must start with 10. But no codevector starts with 10.

(b) We know that $n = 10$ and $k = 9$, so the generator matrix will have 9 rows and 10 columns; hence the first 9 columns will form the identity matrix, and we are left to fill the last column only. Look at the first row of the generator matrix in standard form. It is of the form $100000000*$. It must also be a codevector: its ISBN-10 checksum is $1 \times 1 + 2 \times 0 + 3 \times 0 + \cdots + 9 \times 0 + 10 \times * = 0$, so $1 + 10* = 0$ whence $* = 1$. We similarly deal with the second row, $010000000*$: we have $2 \times 1 + 10* = 0$ so $* = 2$. Continuing in this fashion, we obtain

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 2 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 4 \\ & & & & \ddots & & & & & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 9 \end{bmatrix}.$$

Week 4

Decoding linear codes

Version 2023-10-07. To accessible online version of this chapter

Synopsis. *We explicitly describe a decoder* DECODE: $\mathbb{F}_q^n \to C$ *based on* **coset leaders** *and a* **standard array** *for* C. *For binary* C *sent via a binary symmetric channel, we find the probability* $P_{\text{undetect}}(C)$ *of an undetected transmission error. It is related to the* **weight enumerator** *of* C. *We also find the probability* $P_{\text{corr}}(C)$ *that a codeword is decoded correctly.*

Cosets and coset leaders

It turns out that the following notion is of direct relevance to decoding:

Definition: coset

Given a linear code $C \subseteq \mathbb{F}_q^n$ and a vector $\underline{y} \in \mathbb{F}_q^n$, the **coset** of $\underline{y}$ is the set

$$\underline{y} + C = \{\underline{y} + \underline{c} \mid \underline{c} \in C\}.$$

We recall basic facts about cosets (see for example *Algebraic Structures 1*):

- $C = \underline{0} + C$ is itself a coset. (C is called the *trivial coset*.) Moreover, C is the coset of any codeword $\underline{c} \in C$.
- If $\underline{y}, \underline{z} \in \mathbb{F}_q^n$, then either $\underline{y} + C = \underline{z} + C$ (*if* $\underline{y} - \underline{z} \in C$) or $(\underline{y} + C) \cap (\underline{z} + C) = \varnothing$.
- $\#(\underline{y} + C) = \#C = q^k$.
- There are $\dfrac{\#\mathbb{F}_q^n}{\#C} = q^{n-k}$ distinct cosets.

Thus, the whole space $\mathbb{F}_q^n$ is split (*partitioned*) into q^{n-k} cosets:

$$\mathbb{F}_q^n = C \sqcup (\underline{a}_1 + C) \sqcup \ldots \sqcup (\underline{a}_{q^{n-k}-1} + C).$$

The above is true for any abelian group; but the following is specific to Coding Theory:

Definition: coset leader

A **coset leader** of a coset $\underline{y} + C$ is a vector of minimum weight in $\underline{y} + C$.

Remark: warning — a coset leader may not be unique

There may be more than one coset leader in a coset. However, all coset leaders of a given coset are of the same weight.

Proposition 4.1: the formula for a decoder for a linear code

For a linear code $C \subseteq \mathbb{F}_q^n$, any decoder $\texttt{DECODE}\colon \mathbb{F}_q^n \to C$ satisfies:

$$\forall \underline{y} \in \mathbb{F}_q^n \quad \texttt{DECODE}(\underline{y}) = \underline{y} - \underline{e} \text{ where } \underline{e} \text{ is a coset leader of the coset } \underline{y} + C \text{ of } \underline{y}.$$

Proof. Let $\underline{v} = \texttt{DECODE}(\underline{y})$. Then $\underline{v} \in C$. Put $\underline{e} = \underline{y} - \underline{v}$. Since C is a linear code, $-\underline{v} \in C$, so $\underline{e} = \underline{y} + (-\underline{v}) \in \underline{y} + C$. We have proved that $\underline{e}$ must lie in the coset $\underline{y} + C$.

Vector $\underline{y}$ must be decoded to its nearest neighbour in C, i.e., $d(\underline{y}, \underline{v})$ must be minimised. Yet by Lemma 3.1 $d(\underline{y}, \underline{v}) = w(\underline{y} - \underline{v}) = w(\underline{e})$. Hence the decoder must choose $\underline{e}$ so that $w(\underline{e})$ is minimal in the coset $\underline{y} + C$. By definition, $\underline{e}$ must be a coset leader of $\underline{y} + C$. $\square$

Standard array: construction

We now give a method to construct all cosets and to find one coset leader in each coset.

Definition: standard array

A **standard array** for a linear code $C \subseteq \mathbb{F}_q^n$ is a table with the following properties:

- the table has $|C| = q^k$ columns and q^{n-k} rows;
- each row is a coset;
- the leftmost entry in each row is a coset leader of that row;
- the top row is the trivial coset (i.e., C itself);
- each entry is the sum of the leftmost entry in its row and the top of its column;
- the table contains every vector from $\mathbb{F}_q^n$ exactly once.

We explain how to construct a standard array, using the linear code $C = \{0000, 0111, 1011, 1100\} \subseteq \mathbb{F}_2^4$ as an example.

Row 0 of the standard array: lists all *codevectors* (elements of $C = \underline{0} + C$). They must start from $\underline{0}$, but otherwise the order is arbitrary.

$$0000 \quad 0111 \quad 1011 \quad 1100$$

Row 1: out of vectors not yet listed, choose $\underline{a}_1$ of smallest weight — this guarantees that $\underline{a}_1$ will be a coset leader. Fill in Row 1 by adding $\underline{a}_1$ to each codevector in Row 0.
Say, $\underline{a}_1 = 0001$. To list its coset, add it to row 0: e.g., $0001 + 0111 = 0110$, etc.

$$0001 \quad 0110 \quad 1010 \quad 1101$$

Row 2: choose $\underline{a}_2$ of smallest weight not yet listed, and do the same as for Row 1.
Say, $\underline{a}_2 = 0010$, add it to row 0:

$$0010 \quad 0101 \quad 1001 \quad 1110$$

Row 3: same with, say, $\underline{a}_3 = 0100$:

$$0100 \quad 0011 \quad 1111 \quad 1000$$

Since we have filled 4 rows, and $q^{n-k} = 2^{4-2} = 4$, our standard array is complete:

Example: a standard array for the code $C = \{0000, 0111, 1011, 1100\}$

0000	0111	1011	1100
0001	0110	1010	1101
0010	0101	1001	1110
0100	0011	1111	1000

Standard array: decoding

Let $C \subseteq \mathbb{F}_q^n$ be a linear code. By Proposition 4.1, any decoder is given by

$$\texttt{DECODE}(\underline{y}) = \underline{y} - \texttt{COSET LEADER}(\underline{y} + C).$$

This suggests the following decoding algorithm for C.

Algorithm 4.2: the standard array decoder

Preparation: construct a standard array for C.
Decoding:

- Receive a vector $\underline{y} \in \mathbb{F}_q^n$.
- Look up $\underline{y}$ in the standard array.
- Return the topmost vector of the column of $\underline{y}$ as $\texttt{DECODE}(\underline{y})$.

Justification: the algorithm is correct because, by definition of a standard array,

(a) Look-up of $\underline{y}$ will succeed as every vector in $\mathbb{F}_q^n$ is present in the array;
(b) the row of $\underline{y}$ starts with COSET LEADER$(\underline{y}+C)$, so
(c) the top of $\underline{y}$'s column is $\underline{y}-$ COSET LEADER$(\underline{y}+C)$ so this is $\underline{y}$ decoded.

Example: use the standard array decoder

For the code $C = \{0000, 0111, 1011, 1100\}$ and standard array constructed above,

- decode the received vectors 0011 and 1100;
- give an example of one bit error occurring in a codeword and being corrected;
- give an example of one bit error occurring in a codeword and not being corrected.

Solution. We work with the following standard array for C:

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 0100 & 0011 & 1111 & 1000 \end{array}$$

The received vector 0011 is in the second column, so DECODE$(0011) = 0111$. The received vector 1100 is a codeword (in the fourth column), so DECODE$(1100) = 1100$.

Suppose that the codeword 0000 is sent. If an error occurs in the last bit, the word 0001 is received and decoded correctly as 0000. If an error occurs in the first bit, the word 1000 is received and decoded incorrectly as 1100.

Discussion: is a standard array decoder unique?

Recall that there may be more than one possible standard array for the code C. Indeed, in the above example the coset $0100 + C$ has two coset leaders: 0100 and 1000. Thus, we could construct a different standard array for C:

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 1000 & 1111 & 0011 & 0100 \end{array}$$

The decoder associated to this standard array is different from the decoder considered above. Both decoders decode the same linear code C. A linear code can have more than one decoder.

However, if C is a **perfect** linear code, then each coset has only one coset leader, so the **decoder is unique.** This property of perfect codes appears on the example sheets.

Reminder (the number of errors corrected by a code)

Recall that a code with minimum distance d corrects $t = [(d-1)/2]$ errors.

The code C in the above example is linear, hence $d(C) = w(C) = 2$ (it is easy to find the minimum weight of the code by inspection). This means that the code corrects $\left[\frac{2-1}{2}\right] = 0$ errors. That is, C is not guaranteed to correct even a single bit error occurring in a codevector. And indeed, we saw in an example how one bit error occurred in a codevector and was not corrected.

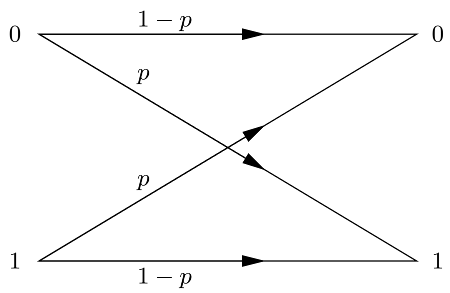
So, from the point of view of Hamming's theory, this code C has no error-correcting capability. It still detects up to one error.

But in Shannon's theory, error-detecting and error-correcting performance of a code are measured probabilistically.

Error-detecting and error-correcting performance of a linear code: Shannon's theory point of view

Shannon's Information Theory is interested in how likely is it that a transmission error in a codeword is not detected/corrected by a decoder of C. We will answer these questions for a binary linear code C transmitted via $BSC(p)$.

Recall that this means that one bit (0 or 1), transmitted via the channel, arrives unchanged with probability $1-p$, and gets flipped with probability p:



When a codeword $\underline{v}$ is transmitted, the channel generates a random error vector and adds it to $\underline{v}$. By definition of $BSC(p)$, for a given $\underline{e} \in \mathbb{F}_2^n$ one has

$$P(\text{the error vector equals } \underline{e}) = (1-p)^{n-i}p^i, \qquad \text{where } i = w(\underline{e}).$$

In determining $P_{\text{undetect}}(C)$, the following notion is very useful:

Definition: the weight enumerator

The **weight enumerator** of a linear code $C \subseteq \mathbb{F}_q^n$ is the polynomial

$$W_C(x,y) = \sum\nolimits_{\underline{v} \in C} x^{n-w(\underline{v})} y^{w(\underline{v})} = A_0 x^n + A_1 x^{n-1} y + A_2 x^{n-2} y^2 + \ldots + A_n y^n$$

in two variables x, y, where $A_i = \#\{\underline{v} \in C : w(\underline{v}) = i\}$.

Theorem 4.3: $P_{\text{undetect}}(C)$, the probability of an undetected error

Suppose that a codevector of a binary linear code C of length n is transmitted via $BSC(p)$. The probability of an undetected error is

$$P_{\text{undetect}}(C) = W_C(1-p, p) - (1-p)^n.$$

Proof. Let $\underline{v} \in C$ be the codevector being transmitted. Recall that an *undetected error* means that the received vector $\underline{v} + \underline{e}$ is a codevector not equal to $\underline{v}$. Note that, since $\underline{v} \in C$ and C is a vector space,

$$\underline{v} + \underline{e} \in C,\ \underline{v} + \underline{e} \neq \underline{v} \quad \iff \quad \underline{e} \in C,\ \underline{e} \neq \underline{0}.$$

Therefore, *an undetected error means that the error vector is a non-zero codevector.* We can now calculate

$$P_{\text{undetect}}(C) = \sum_{\underline{e} \in C,\, \underline{e} \neq \underline{0}} P(\text{the error vector is } \underline{e}) = \sum_{\underline{e} \in C,\, \underline{e} \neq \underline{0}} (1-p)^{n-w(\underline{e})} p^{w(\underline{e})}.$$

This is $W_C(1-p, p)$ without exactly one term, excluded by the constraint $\underline{e} \neq \underline{0}$, namely

$$(1-p)^{n-w(\underline{0})} p^{w(\underline{0})} = (1-p)^n,$$

which gives the expression for $P_{\text{undetect}}(C)$ as stated. □

Remark: In general, $P_{\text{undetect}}(C)$ is calculated assuming that the codeword $\underline{v}$ is picked at random from the code. However, our proof shows that for a *linear* binary code and for the binary *symmetric* channel the probability is the same for all codevectors.

Example: calculating the weight enumerator and P_{undetect}

The binary linear code $C = \{0000, 0111, 1011, 1100\}$ has one codeword of weight 0, zero codewords of weight 1, one codeword of weight 2 and two codewords of weight 3. Hence the weight enumerator of C is

$$W_C(x,y) = x^4 + x^2 y^2 + 2xy^3.$$

If a codeword of C is sent via $BSC(p)$, an undetected error occurs with probability

$$P_{\text{undetect}}(C) = (1-p)^2p^2 + 2(1-p)p^3.$$

Discussion. Knowing $P_{\text{undetect}}(C)$ is useful when a code is used for error detection, e.g., if the receiver can request retransmission if an error is detected. Then $P_{\text{undetect}}(C)$ is on average the proportion of incorrect codevectors, hence incorrect symbols, accepted by the receiver. A code should be designed for a particular channel so as to keep this probability below an agreed threshold.

The probability of correct decoding

We will now find the probability of an error being *corrected* for C.

Theorem 4.4: $P_{\text{corr}}(C)$, the probability of correct decoding

Suppose that a codevector of a binary linear code C is transmitted via $BSC(p)$. The probability that the received vector will be decoded correctly is

$$P_{\text{corr}}(C) = \sum_{i=0}^{n} \alpha_i (1-p)^{n-i}p^i,$$

where α_i denotes the number of cosets where the coset leader is of weight i.

Proof. Recall that $\underline{v} \in C$ is *decoded correctly* if $\texttt{DECODE}(\underline{v}+\underline{e}) = \underline{v}$. By Proposition 4.1,

$$\begin{aligned}\texttt{DECODE}(\underline{v}+\underline{e}) &= \underline{v}+\underline{e} - \texttt{COSET LEADER}(\underline{v}+\underline{e}) \\ &= \underline{v}+\underline{e} - \texttt{COSET LEADER}(\underline{e}).\end{aligned}$$

Therefore, *correct decoding occurs if the error vector is the chosen coset leader of its coset.*

We therefore have one good outcome per coset: namely, $\underline{e}$ equals the chosen coset leader of the coset. Recall that that happens with probability $(1-p)^{n-i}p^i$ where i is the weight of the coset leader of the given coset. Summing over all cosets and gathering the like terms, we obtain the formula for $P_{\text{corr}}(C)$ as stated (it does not depend on $\underline{v}$). $\square$

Discussion. When is it important to know $P_{\text{corr}}(C)$? In one-way communication channels without retransmission even if an error is detected, the decoder produces a best guess as to which codevector was sent. An example is computer memory, where information could have been written ("sent") long time ago, and it is not possible to "resend" it. Thus, $1 - P_{\text{corr}}(C)$ is on average the proportion of incorrect codevectors, hence incorrect symbols, accepted by the receiver. A code should be designed for a particular channel to keep $1 - P_{\text{corr}}(C)$ below an agreed threshold.

Example: calculation of P_{corr}

Let $C = \{0000, 0111, 1011, 1100\}$. From the standard array

$$\begin{array}{cccc} 0000 & 0111 & 1011 & 1100 \\ 0001 & 0110 & 1010 & 1101 \\ 0010 & 0101 & 1001 & 1110 \\ 1000 & 1111 & 0011 & 0100 \end{array}$$

we can see that $\alpha_0 = 1$, $\alpha_1 = 3$, $\alpha_2 = \alpha_3 = 0$ (given by the leftmost column), so

$$P_{\text{corr}}(C) = (1-p)^4 + 3(1-p)^3 p.$$

Comparing codes using approximate values of P_{undetect} or P_{corr}

Our analysis above gives, for a binary linear code C transmitted via $BSC(p)$, the probabilities $P_{\text{undetect}}(C)$ and $P_{\text{corr}}(C)$ as polynomials in p.

In practical situations p is typically very small, so it is rarely useful to know all terms of these polynomials in p. The term with the lowest power of p will dominate and can be used as an approximate value of the probability. This makes comparing codes easier.

Example. We compare use of the code C against transmission of undencoded information ("trivial binary code of length 1").

If C is used for error detection: $P_{\text{undetect}}(C) = (1-p)^2 p^2 + 2(1-p)p^3$. This is a polynomial of the form $p^2 + o(p^2)$ where $o(p^2)$ contains powers of p higher than 2. For small p, the terms in $o(p^2)$ are negligible compared to p^2. We conclude:

$$P_{\text{undetect}}(C) \sim p^2 \text{ whereas } P_{\text{undetect}}(\text{no encoding}) = p,$$

i.e., the use of C improves the proportion of bad bits in the output from p to p^2.

If C is used for error correction: $1 - P_{\text{corr}}(C) = 1 - (1-p)^4 - 3(1-p)^3 p = p + o(p)$ (check this by opening the brackets). Thus,

$$1 - P_{\text{corr}}(C) \sim p \text{ whereas } 1 - P_{\text{corr}}(\text{no encoding}) = p.$$

Hence C is useless for error correction: its use does not improve the proportion of bad bits in the output while increasing the volume of information transmitted two-fold ($R = 0.5$).

The above suggests that for error correction, more mathematically sophisticated codes need to be designed. In the rest of the course, we will see how ideas from different areas of mathematics are used in code constructions.

Week 4

Exercises (answers at end)

Version 2023-11-04. To accessible online version of these exercises

Exercise 4.1. Write down the weight enumerator of $Rep(n, \mathbb{F}_2)$, more generally of $Rep(n, \mathbb{F}_q)$.

Notation: below, $C \subseteq \mathbb{F}_q^n$ is a linear code, $d(C) = d$, and $t = \left[\frac{d-1}{2}\right]$.

Exercise 4.2. Prove that each vector $\underline{a}$ of weight $\leq t$ in the space $\mathbb{F}_q^n$ is a **unique coset leader** (that is, $w(\underline{a})$ is **strictly** less than weights of all other vectors in its coset $\underline{a} + C$).

Hint. If $\underline{a} \neq \underline{b}$ are in the same coset, show that $d \leq w(\underline{a}) + w(\underline{b})$. Then use $d - t > t$.

Exercise 4.3 (important fact about perfect linear codes — needed for exam)**.** Assume C is perfect. Use the Hamming bound to show that the number of cosets equals $\#S_t(\underline{0})$, i.e., there as many cosets as vectors of weight $\leq t$ in the space $\mathbb{F}_q^n$. Deduce that every coset has a unique coset leader, and that the coset leaders are exactly the vectors of weight $\leq t$.

Exercise 4.4 (not done in tutorial)**.** Find standard arrays for binary codes with each of the following generator matrices. For each code, determine whether every coset has a unique coset leader (i.e., if there is exactly one coset leader in each coset). Find the probability of an undetected / uncorrected error for $BSC(p)$ and argue whether the code is worth using for this channel, compared to transmitting unencoded information.

$$G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad G_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \qquad G_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Exercise 4.5 (more weight enumerators — not done in tutorial)**.** (a) As usual, let $W_C(x, y)$ denote the weight enumerator of a q-ary linear code C. Show that $W_C(1, 0) = 1$ and that $W_C(1, 1) = q^k$ where $k = \dim C$.

(b) Show that the weight enumerator of the trivial binary code $\mathbb{F}_2^n$ is $W_{\mathbb{F}_2^n}(x, y) = (x + y)^n$. Can you write $W_{\mathbb{F}_q^n}(x, y)$ in a similar form?

(c) Write down $W_{E_3}(x, y)$. Can you suggest a compact way to write $W_{E_n}(x, y)$?

Week 4

Exercises — solutions

Version 2023-11-04. To accessible online version of these exercises

Exercise 4.1. Write down the weight enumerator of $Rep(n, \mathbb{F}_2)$, more generally of $Rep(n, \mathbb{F}_q)$.

Answer to E4.1. $Rep(n, \mathbb{F}_2)$ has one codevector of weight 0 and one codevector of weight n. Hence $W_{Rep(n,\mathbb{F}_2)}(x, y) = x^n + y^n$.

Exercise: show that $W_{\mathsf{Rep}(n,\mathbb{F}_q)}(x, y) = x^n + (q-1)y^n$.

Notation: below, $C \subseteq \mathbb{F}_q^n$ is a linear code, $d(C) = d$, and $t = \left[\frac{d-1}{2}\right]$.

Exercise 4.2. Prove that each vector $\underline{a}$ of weight $\le t$ in the space $\mathbb{F}_q^n$ is a **unique coset leader** (that is, $w(\underline{a})$ is **strictly** less than weights of all other vectors in its coset $\underline{a} + C$).

Hint. If $\underline{a} \neq \underline{b}$ are in the same coset, show that $d \le w(\underline{a}) + w(\underline{b})$. Then use $d - t > t$.

Answer to E4.2. If $\underline{a}, \underline{b}$ are in the same coset, then by properties of cosets, $\underline{c} := \underline{a} - \underline{b}$ is a codevector. If $\underline{a} \neq \underline{b}$ then $\underline{c} \neq 0$ and so $d \le w(\underline{c}) = w(\underline{a} - \underline{b}) = d(\underline{a}, \underline{b})$. By the triangle inequality, $d(\underline{a}, \underline{b}) \le d(\underline{a}, \underline{0}) + d(\underline{0}, \underline{b}) = w(\underline{a}) + w(\underline{b})$. Thus, $d \le w(\underline{a}) + w(\underline{b})$ as claimed.

Now assume $w(\underline{a}) \le t$. Then $w(\underline{b}) \ge d - w(\underline{a}) \ge d - t$. But $t < \frac{d}{2}$ so $d - t > t$. We have $w(\underline{b}) \ge d - t > t \ge w(\underline{a})$. This shows that $\underline{a}$ has strictly minimal weight among the vectors in its coset, and so is the unique coset leader.

Exercise 4.3 (important fact about perfect linear codes — needed for exam)**.** Assume C is perfect. Use the Hamming bound to show that the number of cosets equals $\#S_t(\underline{0})$, i.e., there as many cosets as vectors of weight $\le t$ in the space $\mathbb{F}_q^n$. Deduce that every coset has a unique coset leader, and that the coset leaders are exactly the vectors of weight $\le t$.

Answer to E4.3. By the previous exercise, the vectors $\underline{a} \in S_t(\underline{0})$ are unique coset leaders of $\#S_t(\underline{0})$ distinct cosets. The total number of cosets is $\dfrac{q^n}{\#C}$.

Now if C is perfect, then $\#C = \dfrac{q^n}{\#S_t(\underline{0})}$ (the right-hand side is the Hamming bound), and so $\dfrac{q^n}{\#C} = \#S_t(\underline{0})$. Thus if C is perfect, cosets with a unique coset leader of weight $\leq t$ exhaust all cosets, as claimed.

Exercise 4.4 (not done in tutorial)**.** Find standard arrays for binary codes with each of the following generator matrices. For each code, determine whether every coset has a unique coset leader (i.e., if there is exactly one coset leader in each coset). Find the probability of an undetected / uncorrected error for $BSC(p)$ and argue whether the code is worth using for this channel, compared to transmitting unencoded information.

$$G_1 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \qquad G_2 = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}, \qquad G_3 = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Answer to E4.4. G_1 generates the trivial binary code of length 2. Because the code is the whole space $\mathbb{F}_2^2$, its standard array consists of one row:

$$00 \quad 01 \quad 10 \quad 11$$

(the order of the codevectors after 00 is arbitrary). The only coset is the trivial coset which has only one coset leader, 00.

G_2 generates E_3, the even weight code of length 3. It has 4 codevectors and 2 cosets:

000	101	011	110
001	100	010	111

Note that the non-trivial coset has three coset leaders; any of them could be put in column 1.

G_3: list all the 4 codevectors and then use the algorithm for constructing the standard array. One possible answer is given below:

00000	10110	01011	11101
10000	00110	11011	01101
01000	11110	00011	10101
00100	10010	01111	11001
00010	10100	01001	11111
00001	10111	01010	11100
11000	01110	10011	00101
01100	11010	00111	10001

Coset leaders of weight 0 and 1 are the only coset leaders in their cosets. Coset leaders of weight 2 are not unique: e.g., 11000 and 00101 are coset leaders of the same coset.

Error probabilities. The code generated by G_1 is the trivial code, so using it is the same as sending unencoded information.

The code generated by G_2 has weight enumerator $W_{E_3}(x,y) = x^3 + 3xy^2$. Hence an undetected error occurs with probability

$$P_{\text{undetect}}(E_3) = W_{E_3}(1-p,p) - (1-p)^3 = 3(1-p)p^2 \sim 3p^2.$$

Note that this is of the same order as p^2 but at a rate of $2/3$ (recall the code considered in the chapter with worse rate $1/2$).

The probability of an uncorrected error here is $1-P_{\text{corr}}(E_3) = 1-(\alpha_0(1-p)^3+\alpha_1 p(1-p)^2)$ where $\alpha_0 = 1$ (one coset leader of weight 0) and $\alpha_1 = 1$ (one coset leader of weight 1) . We have $1-P_{\text{corr}}(E_3) = 1-((1-p)^3+p(1-p)^2) = 1-(1-p+p)(1-p)^2 = 1-(1-p)^2 \sim 2p$.

The code E_3 does not improve the probability of incorrect decoding. Indeed, Hamming's theory says that E_3 has no error-correcting capability and can only be used for error detection.

The code generated by G_3 has weight enumerator $x^5 + 2x^2y^3 + xy^4$. Hence

$$P_{\text{undetect}} = 2(1-p)^2p^3 + (1-p)p^4 \sim 2p^3.$$

If $p = 0.01$, this is $\approx 2 \times 10^{-6}$, which is 5,000 times better than without encoding.

Furthermore, looking at the coset leaders, we find one coset leader of weight 0, $\alpha_0 = 1$; five coset leaders of weight 1, $\alpha_1 = 5$; two coset leaders of weight 2, $\alpha_2 = 2$. This gives

$$\begin{aligned}
1 - P_{\text{corr}} &= 1 - (\alpha_0(1-p)^5 + \alpha_1 p(1-p)^4 + \alpha_2 p^2(1-p)^3) \\
&= 1 - ((1-p)^2 + 5p(1-p) + 2p^2)(1-p)^3 \\
&= 8p^2 - 14p^3 + 9p^4 - 2p^5 \sim 8p^2.
\end{aligned}$$

If $p = 0.01$, incorrect decoding occurs with probability $\approx 8 \times 10^{-4}$, which is 12.5 times better than without encoding.

Of course, this improvement in reliability comes at a price: the rate of the code is only 0.4, meaning that we have to transmit 2.5 times as much information.

Exercise 4.5 (more weight enumerators — not done in tutorial)**.** (a) As usual, let $W_C(x,y)$ denote the weight enumerator of a q-ary linear code C. Show that $W_C(1,0) = 1$ and that $W_C(1,1) = q^k$ where $k = \dim C$.

(b) Show that the weight enumerator of the trivial binary code $\mathbb{F}_2^n$ is $W_{\mathbb{F}_2^n}(x,y) = (x+y)^n$. Can you write $W_{\mathbb{F}_q^n}(x,y)$ in a similar form?

(c) Write down $W_{E_3}(x,y)$. Can you suggest a compact way to write $W_{E_n}(x,y)$?

Answer to E4.5. (a) Recall $W_C(x,y) = \sum_{\underline{c}\in C} x^{n-w(\underline{c})}y^{w(\underline{c})}$. If $y = 0$, the only non-zero term in this sum is the term without y which corresponds to the (unique) zero codevector of the linear code C; thus, $W_C(x,0) = x^n$ and $W_C(1,0) = 1$. Also, $W_C(1,1) = \sum_{\underline{c}\in C} 1 = \#C = q^k$.

(b) To work out $W_{\mathbb{F}_q^n}(x,y)$, write it in the form $W_{\mathbb{F}_q^n}(x,y) = \sum_{i=0}^n A_i x^{n-i} y^i$ where $A_i = \#\{\underline{v} \in \mathbb{F}_q^n : w(\underline{v}) = i\}$. Note that $w(\underline{v}) = d(\underline{v}, \underline{0})$, and in the proof of the Hamming bound we calculated the number of words at distance i from $\underline{0}$ (or from any other fixed vector) to be $\binom{n}{i}(q-1)^i$. Hence

$$W_{\mathbb{F}_q^n}(x,y) = \sum_{i=0}^{n} \binom{n}{i} (q-1)^i x^{n-i} y^i = (x + (q-1)y)^n.$$

(c) The even weight code E_3 is $\{000, 011, 101, 110\}$, so that $W_{E_3}(x,y) = x^3 + 3xy^2$. The weight enumerator of E_n will be obtained in the lectures as an application of the MacWilliams identity.

Week 5

The dual code. Syndrome decoding

Version 2023-11-01. To accessible online version of this chapter

Synopsis. *Every linear code C has a dual code, $C^{\perp}$, and check matrices. While a generator matrix G is used to encode messages into codevectors, a check matrix H serves to detect errors — and to correct them using syndrome decoding.*

Motivation. The inner product of vectors

Let C be a linear code. Given a received vector $\underline{y}$, how to test whether $\underline{y} \in C$? Storing all codevectors of C is not an option for codes of large length and dimension, whose use is dictated by modern applications to low-noise channels. Storing just a generator matrix G of C is better in terms of storage space, but testing whether $\underline{y}$ is in the row space of G can be computationally demanding.

Some codes, however, are defined by a single *checksum* — recall the even weight code and the ISBN-10 code. A checksum of a given vector is easy to compute.

Extending the checksum approach, we introduce a *check matrix* which generates the *dual code*. It turns out that this construction helps to correct errors as well (not just detect). The first notion we need is:

Definition: inner product

For $\underline{u}, \underline{v} \in \mathbb{F}_q^n$, the scalar (element of $\mathbb{F}_q$) defined as $\underline{u} \cdot \underline{v} = \sum_{i=1}^{n} u_i v_i$ is called the **inner product** of the vectors $\underline{u}$ and $\underline{v}$.

Example: some inner products in $\mathbb{F}_2^3$

For $111, 101 \in \mathbb{F}_2^3$, one has

$111{\cdot}111 = 1^2{+}1^2{+}1^2 = 1,\ 111{\cdot}101 = 1{\cdot}1{+}1{\cdot}0{+}1{\cdot}1 = 0,\ 101{\cdot}101 = 1^2{+}0^2{+}1^2 = 0.$

If $C \subset \mathbb{F}_q^n$ is a set, $\underline{v} \in \mathbb{F}_q^n$, we may write $\underline{v} \cdot C$ to denote the set $\{\underline{v} \cdot \underline{c} \mid \underline{c} \in C\}$.

Properties of the inner product

(1) **Expression as a matrix product:** $\underline{u} \cdot \underline{v} = \underline{u}\,\underline{v}^T$.

Explanation: we write elements of $\mathbb{F}_q^n$ as row vectors. Thus, $\underline{u}$ is a row vector $(u_1, \ldots, u_n)$, and $\underline{v}^T$ is the transpose of $\underline{v}$, so a column vector $\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$. Multiplying $\underline{u}$, an $1 \times n$ matrix, and $\underline{v}^T$, an $n \times 1$ matrix, we obtain a 1×1 matrix, which we identify with a scalar in $\mathbb{F}_q$.

(2) **Symmetry:** $\underline{u} \cdot \underline{v} = \underline{v} \cdot \underline{u}$. (**Explanation:** this is easily seen from the definition.)

(3) **Bilinearity:** for a scalar $\lambda \in \mathbb{F}_q$ we have $(\underline{u} + \lambda\underline{w}) \cdot \underline{v} = \underline{u} \cdot \underline{v} + \lambda(\underline{w} \cdot \underline{v})$ and $\underline{u} \cdot (\underline{v} + \lambda w) = \underline{u} \cdot \underline{v} + \lambda(\underline{u} \cdot \underline{w})$. (**Explanation:** from linear algebra, the matrix product in $\underline{u}\,\underline{v}^T$ is bilinear.)

(4) **Non-degeneracy:** $\underline{u} \cdot \mathbb{F}_q^n = \{0\}$, if and only if $\underline{u} = 0$.

Explanation: let $\underline{\epsilon_i} = (0, \ldots, 0, 1, 0, \ldots, 0)$ be the vector with ith symbol 1 and all other symbols 0. Then $\underline{u} \cdot \underline{\epsilon_i} = u_i$. So if $\underline{u} \cdot \mathbb{F}_q^n = \{0\}$, then in particular $\underline{u} \cdot \underline{\epsilon_i} = 0$ hence $u_i = 0$, for all i, meaning that $\underline{u}$ is the zero vector. And if $\underline{u} = \underline{0}$, then $\underline{u} \cdot \underline{c} = 0$ for all $\underline{c} \in \mathbb{F}_q^n$.

The dual code

Definition: dual code

Given a code $C \subseteq \mathbb{F}_q^n$, we define the **dual code** $C^\perp$ as

$$C^\perp = \{\underline{v} \in \mathbb{F}_q^n \mid \underline{v} \cdot C = \{0\}\}.$$

We can say that $C^\perp$ consists of all vectors **orthogonal** to the code C (where $\underline{v}$ orthogonal to C means $\underline{v} \cdot C = \{0\}$).

Exercise. Using bilinearity of the inner product, show that $C^\perp$ is a *linear* code.

Recall that $Rep(n, \mathbb{F}_2) = \{00\ldots0, 11\ldots1\} \subseteq \mathbb{F}_2^n$ is the binary repetition code of length n. We now work out the code $Rep(n, \mathbb{F}_2)^\perp$ using the definition.

By definition, $Rep(n, \mathbb{F}_2)^\perp = \{\underline{v} \in \mathbb{F}_2^n \mid \underline{v} \cdot 00\ldots0 = 0,\ \underline{v} \cdot 11\ldots1 = 0\}$. The first condition, $\underline{v} \cdot \underline{0} = 0$ is vacuous (holds for all vectors $\underline{v} \in \mathbb{F}_2^n$). The second condition, $\underline{v} \cdot 11\ldots1$, means $v_1 + v_2 + \ldots + v_n = 0$ in $\mathbb{F}_2$, i.e., $\underline{v} \in E_n$, the binary even weight code of length n. Thus:

Example: the dual code of the binary repetition code

$Rep(n, \mathbb{F}_2)^\perp = E_n$.

Check matrices

Definition: check matrix

A **check matrix** for a linear code C means a generator matrix for $C^\perp$.

One sometimes says *parity check matrix* (the term arose from applications of binary codes).

Theorem 5.1: properties of the dual code and a check matrix

If $C \subseteq \mathbb{F}_q^n$ is a linear code of dimension k, then:

i. $\dim C^\perp = n - k$;

ii. $C = \{\underline{v} \in \mathbb{F}_q^n : \underline{v}H^T = \underline{0}\}$ for any check matrix H of C.

Proof. We recall the *Rank-Nullity Theorem* from Linear Algebra: if M is a matrix with n columns, then

$$\operatorname{rank}(M) + \dim \operatorname{Nullspace}(M) = n,$$

where $\operatorname{rank}(M)$ is the dimension of the span of the rows of M, and $\operatorname{Nullspace}(M)$ can be written as $\{\underline{v} \in \mathbb{F}_q^n : M\underline{v}^T = \overline{0}\}$.

i. Consider the matrix $\Big[C\Big]$ made up of *all* codevectors of C used as rows. The $\operatorname{Nullspace}(\Big[C\Big])$ is the set $\{\underline{v} : \Big[C\Big]\,\underline{v}^T = \overline{0}\}$. Note that the column vector $\Big[C\Big]\,\underline{v}^T$ is $\begin{bmatrix} \underline{c}_1\underline{v}^T \\ \underline{c}_2\underline{v}^T \\ \vdots \end{bmatrix} = \begin{bmatrix} \underline{c}_1 \cdot \underline{v} \\ \underline{c}_2 \cdot \underline{v} \\ \vdots \end{bmatrix}$, which is zero if and only if the inner product $\underline{c} \cdot \underline{v}$ is 0 for all rows $\underline{c}$ of $\Big[C\Big]$, i.e., for all codevectors $\underline{c}$ of C. By definition of the dual code, this happens exactly when $\underline{v} \in C^\perp$, so $\operatorname{Nullspace}(\Big[C\Big]) = C^\perp$. By rank-nullity, $\dim C^\perp = n - \operatorname{rank}(\Big[C\Big])$. Since the rows of $\Big[C\Big]$ span C, one has $\operatorname{rank}(\Big[C\Big]) = \dim C = k$ and so $\dim C^\perp = n - k$.

ii. By definition H generates the code $C^\perp$; so by i., H has $n-k$ rows, $H = \begin{bmatrix} \underline{r}_1 \\ \vdots \\ \underline{r}_{n-k} \end{bmatrix}$. Thus, $\operatorname{rank}(H) = \dim C^\perp = n - k$, and so by rank-nullity, $\dim \operatorname{Nullspace}(H) = n - (n-k) = k$. Note that $C \subseteq \operatorname{Nullspace}(H)$: indeed, if $\underline{c} \in C$, then $\underline{r}_i\underline{c}^T = \underline{r}_i \cdot \underline{c} = 0$ for all i because $\underline{r}_i \in C^\perp$, which means that $H\underline{c}^T = \overline{0}$. Since $\dim C = \dim \operatorname{Nullspace}(H)$, it follows that $C = \operatorname{Nullspace}(H)$, which is $\{\underline{v} : H\underline{v}^T = \overline{0}\}$.

The law $(AB)^T = B^T A^T$ for the product of matrices implies that $(\underline{v}H^T)^T = H\underline{v}^T$, and so $H\underline{v}^T$ is zero iff $\underline{v}H^T$ is. Thus, $C = \{\underline{v} \in \mathbb{F}_q^n : \underline{v}H^T = \underline{0}\}$ as claimed. $\square$

The syndrome of a vector

Definition: syndrome

Let H be a check matrix for a linear code $C \subseteq \mathbb{F}_q^n$. Let $\underline{y} \in \mathbb{F}_q^n$. The vector

$$S(\underline{y}) = \underline{y}H^T$$

is called the **syndrome** of $\underline{y}$. The linear map $S \colon \mathbb{F}_q^n \to \mathbb{F}_q^{n-k}$ is the **syndrome map.**

Proposition 5.2: syndromes of vectors in the same coset

Let S be a syndrome map for a linear code $C \subseteq \mathbb{F}_q^n$. If $\underline{v}, \underline{y} \in \mathbb{F}_q^n$,

- $S(\underline{v}) = S(\underline{y}) \iff \underline{v}, \underline{y}$ are in the same coset of C;
- $S(\underline{v}) = \underline{0} \iff \underline{v} \in C$.

Proof. $\underline{y}H^T = \underline{v}H^T \iff (\underline{y} - \underline{v})H^T = \underline{0} \iff \underline{y} - \underline{v} \in C$. By definition of cosets, this means that $\underline{v}$ is in the coset of $\underline{y}$. In particular, $S(\underline{v}) = \underline{0}$ means $\underline{v} \in \underline{0} + C = C$. □

The use of syndromes in error detection and correction

If $S(\underline{y}) \neq 0$, $\underline{y}$ is not a codevector, so the syndrome map *detects* errors in a received vector.

To *correct* errors, we need to construct a decoder for the linear code C. If we know a check matrix H for C, we can improve the standard array decoder for C. We will write the same decoder differently; it will require much less memory but more calculations.

Algorithm 5.3: the syndrome decoder

Preparation. Construct a *table of syndromes*, with q^{n-k} rows, of the form

Coset leader $\underline{a}_i$	$S(\underline{a}_i)$

Start with the top row: the codeword $\underline{0}$ and its syndrome $S(\underline{0}) = \underline{0}$.
At each step, choose a vector $\underline{a}_i \in \mathbb{F}_q^n$ of smallest weight such that $S(\underline{a}_i)$ does not appear in the table; then $\underline{a}_i$ is a coset leader of a new coset.
Decoding.

- Receive a vector $\underline{y} \in \mathbb{F}_q^n$.
- Calculate $S(\underline{y}) = \underline{y}H^T$.

- In the table, find $\underline{a}_i$ with $S(\underline{a}_i) = S(\underline{y})$. Then $\underline{a}_i$ is the coset leader of $\underline{y} + C$.
- Return $\mathtt{DECODE}(\underline{y}) = \underline{y} - \underline{a}_i$.

Remark. The syndrome decoder is based on a choice of one coset leader in every coset. This is the same as for the standard array decoder.

In fact, if the same coset leaders are chosen in both decoders, both decoders with yield *the same* function $\mathtt{DECODE}\colon \mathbb{F}_q^n \to C$. They differ only in the way this function is computed.

The number of arithmetic operations required to calculate the syndrome $S(\underline{y}) = \underline{y}H^T$ can be of order n^2, whereas the standard array decoder requires $\sim n$ operations to look up a vector. On the other hand, the amount of memory required by the syndrome decoder is proportional to q^{n-k} which is better than q^n for the standard array. The advantage is especially significant for codes with high code rate $\dfrac{k}{n}$.

Nevertheless, for codes which have more algebraic structure (than just linear codes), decoding algorithms exist which require even less storage, but the computation complexity is higher compared to syndrome decoding. Some examples will appear from the next chapter onwards.

Example: example of syndrome decoding

Let C be the binary linear code with check matrix $H = \left[\begin{array}{cc|cccc} 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \end{array}\right]$.

(a) Construct the table of syndromes for C using the matrix H.

(b) Using the table of syndromes, decode the received vector $\underline{y} = 111111$.

Solution.

(a) When calculating syndromes, it is useful to observe that the syndrome of a vector $0\ldots010\ldots0$ (with 1 in position i and 0s elsewhere) is equal to the ith column of H, transposed.

The syndrome map is linear, so the syndrome of a sum of two vectors is the sum of their syndromes, etc.

For example, $S(011000) = 0011 + 1000 = 1011$ (the sum of the second and the third columns of H, transposed).

vector	syndrome	leader?
000000	0000	yes
000001	0001	yes
000010	0010	yes
000100	0100	yes
001000	1000	yes
010000	0011	yes
100000	0110	yes

All vectors of weight 1 have different syndromes, so they all are coset leaders. We need more coset leaders, hence we start looking at vectors of weight 2, then weight 3:

000011	0011	no, syndrome already in the table
000101	0101	yes
001001	1001	yes
001010	1010	yes
001100	1100	yes
010100	0111	yes
011000	1011	yes
101000	1110	yes
001101	1101	yes
011100	1111	yes

When we try a vector, say of weight 2, and find that is syndrome is already in the table, we ignore that vector and try another one.

We found $16 = 2^{6-2}$ coset leaders so we stop.

(b) $S(111111) = 1010$ which is the syndrome of the coset leader 001010 in the table. Therefore, $\texttt{DECODE}(111111) = 111111 - 001010 = 110101$.

Week 5

Exercises (answers at end)

Version 2023-10-24. To accessible online version of these exercises

Exercise 5.1. Let C be a linear code of length n and weight 1. Show: $C^{\perp}$ has no codevectors of weight n.

Exercise 5.2. Use Theorem 5.1 to show that $(C^{\perp})^{\perp} = C$ for every linear code C.

Exercise 5.3. A linear code C is **self-orthogonal**, if $\forall \underline{v}, \underline{w} \in C$, $\underline{v} \cdot \underline{w} = 0$; equivalently, $C \subseteq C^{\perp}$.

(a) Let G be a generator matrix for C. Show: C self-orthogonal $\Leftrightarrow GG^T = \mathbf{0}$ (zero matrix).

(b) Which of the following codes are self-orthogonal: $Rep(n, \mathbb{F}_2)$, E_n, the ternary code generated by $G = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 1 & 1 \end{bmatrix}$?

Exercise 5.4. A linear code C is called **self-dual** if $C = C^{\perp}$. (Clearly, a self-dual code is self-orthogonal.)

(a) Show: a linear $[n, k, d]_q$-code C is self-dual $\iff$ C is self-orthogonal and $k = n/2$. Deduce that self-dual codes have even length.

(b) [*2013 exam*] Show that a binary code generated by $\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$ is self-dual.

Exercise 5.5 (not done in tutorial)**.** (a) [*2013 exam, B6b*] Show that *binary* self-orthogonal codes have even weight. *Hint*: if $\underline{c} \in C$, what is $\underline{c} \cdot \underline{c}$?

(b) [*2016 B5*] Show: *ternary* self-orthogonal codes have weight divisible by 3. (Hint as in (a).)

(c) [*2015 B4*] Prove that for every even n there exists a 5-ary self-dual code of length n. (*Hint*: look for a matrix.)

Week 5

Exercises — solutions

Version 2023-10-24. To accessible online version of these exercises

Exercise 5.1. Let C be a linear code of length n and weight 1. Show: $C^\perp$ has no codevectors of weight n.

Answer to E5.1. Let $\underline{c} = (0, \dots, 0, \lambda, 0, \dots, 0)$ be a vector of weight 1 in C, where the only non-zero symbol λ is in ith position. Then for any $\underline{x} = (x_1, \dots, x_n) \in C^\perp$ one has $\underline{x} \cdot \underline{c} = 0$ which reads $\lambda x_i = 0$. Because $\lambda \neq 0$, this is equivalent to $x_i = 0$. Hence the weight of $\underline{x}$ is at most $n - 1$, because at least one symbol in $\underline{x}$ is zero.

Exercise 5.2. Use Theorem 5.1 to show that $(C^\perp)^\perp = C$ for every linear code C.

Answer to E5.2. Let $C \subseteq \mathbb{F}_q^n$ be a linear code of dimension k. Take $\underline{c} \in C$ and $\underline{v} \in C^\perp$. By definition of the dual code, $\underline{c} \cdot \underline{v} = 0$. Since this is true for all $\underline{v} \in C^\perp$, by definition of the dual code again, $\underline{c} \in (C^\perp)^\perp$. We proved that $C \subseteq (C^\perp)^\perp$. Since $\dim C = k = n - (n - k)$ which is $\dim (C^\perp)^\perp$ by Theorem 5.1, one has $C = (C^\perp)^\perp$.

Exercise 5.3. A linear code C is **self-orthogonal**, if $\forall \underline{v}, \underline{w} \in C$, $\underline{v} \cdot \underline{w} = 0$; equivalently, $C \subseteq C^\perp$.

(a) Let G be a generator matrix for C. Show: C self-orthogonal $\Leftrightarrow GG^T = \mathbf{0}$ (zero matrix).

(b) Which of the following codes are self-orthogonal: $Rep(n, \mathbb{F}_2)$, E_n, the ternary code generated by $G = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 1 & 1 \end{bmatrix}$?

Answer to E5.3.

(a) Recall that $C = \{\underline{u}G : \underline{u} \in \mathbb{F}_q^k\}$. So C is self-orthogonal iff $\forall \underline{u}, \underline{v} \in \mathbb{F}_q^k$, $(\underline{u}G) \cdot (\underline{v}G) = \underline{u}GG^T\underline{v}^T = 0$. The latter is true iff GG^T is the $k \times k$ zero matrix (because, by taking $\underline{u}, \underline{v}$ to be vector units, $\underline{u}GG^T\underline{v}^T$ can be made equal to each entry of GG^T).

(b) The repetition code $Rep(n, \mathbb{F}_2)$ has generator matrix $G = [11\ldots1]$ (a single row of n ones). We have $GG^T = 11\ldots1 \cdot 11\ldots1 = 1 + 1 + \cdots + 1$ (n times) which is zero if n is even and is 1 if n is odd. So $Rep(n, \mathbb{F}_2)$ is self-orthogonal iff n is even.

Consider the vectors $\underline{u} = 1100\ldots00$ and $\underline{v} = 0110\ldots00$, of weight 2, in the even weight code E_n. Note that the inner product $\underline{u} \cdot \underline{v}$ is 1. Therefore, E_n is not self-orthogonal... The only gap in this argument is that to construct such vectors $\underline{u}$ and $\underline{v}$, we need $n \geq 3$. And if $n = 2$, $E_2 = \{00, 11\} = Rep(2, \mathbb{F}_2)$ is self-orthogonal. If $n \geq 3$, then E_n is not self-orthogonal.

Finally, the ternary code has $GG^T = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$ (check this!) and so is self-orthogonal.

Exercise 5.4. A linear code C is called **self-dual** if $C = C^\perp$. (Clearly, a self-dual code is self-orthogonal.)

(a) Show: a linear $[n, k, d]_q$-code C is self-dual $\iff$ C is self-orthogonal and $k = n/2$. Deduce that self-dual codes have even length.

(b) [*2013 exam*] Show that a binary code generated by $\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$ is self-dual.

Answer to E5.4. (a) Self-orthogonal is equivalent to $C \subseteq C^\perp$. Self-dual means $C = C^\perp$; this is equivalent to $C \subseteq C^\perp$ AND $\dim C = \dim C^\perp$. The equality of dimensions reads $k = n - k$, i.e., $n = 2k$. Thus, the length of a self-dual code is twice its dimension; in particular, the length is even.

(b) The code is self-orthogonal because $G = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$ satisfies $GG^T = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$. We have $n = 2k$ as $4 = 2 \times 2$ so by (a) the code is self-dual.

Exercise 5.5 (not done in tutorial)**.** (a) [*2013 exam, B6b*] Show that *binary* self-orthogonal codes have even weight. *Hint*: if $\underline{c} \in C$, what is $\underline{c} \cdot \underline{c}$?

(b) [*2016 B5*] Show: *ternary* self-orthogonal codes have weight divisible by 3. (Hint as in (a).)

(c) [*2015 B4*] Prove that for every even n there exists a 5-ary self-dual code of length n. (*Hint*: look for a matrix.)

Answer to E5.5. (a) Let $C \subseteq \mathbb{F}_2^n$ be a self-orthogonal binary code. Then in particular, $\underline{c} \cdot \underline{c} = 0$ for all $\underline{c} \in C$. The inner product $\underline{c} \cdot \underline{c}$ rewrites as $c_1^2 + c_2^2 + \cdots + c_n^2 = 0$. Note that in the field $\mathbb{F}_2$ one has $x^2 = x$ for all x. Hence $c_1 + c_2 + \cdots + c_n = 0$ for all $\underline{c} \in C$. But this says that the vector $\underline{c}$ is of even weight. We proved that if binary C is self-orthogonal, then all codevectors of C are of even weight. In particular, $w(C)$, the minimum positive weight of a codevector of C, is even.

(b) Let $\underline{c} \in C$ be the vector of weight $w(C)$. Since C is self-orthogonal, $\underline{c} \cdot \underline{c} = 0$ in $\mathbb{F}_3$. Note that $\underline{c} \cdot \underline{c} = c_1^2 + c_2^2 + \cdots + c_n^2$ is obtained by adding up the squares of all non-zero symbols in $\underline{c}$. But the square of any non-zero symbol is 1: $1^2 = 2^2 = 1$ in $\mathbb{F}_3$. Hence $\underline{c} \cdot \underline{c}$ is equal to the sum of $w(C)$ 1s in $\mathbb{F}_3$; thus, $w(C)$ is zero in $\mathbb{F}_3$, meaning that $w(C)$ is a multiple of 3.

(c) Consider the matrix $G = \left[\begin{array}{c|c} I_k & 2I_k \end{array} \right]$ over the field $\mathbb{F}_5$, where $n = 2k$. Note that $GG^T = I_k^2 + 4I_k^2 = 5I_k^2 = 0$ so the code C, generated by the matrix G, is self-orthogonal. Moreover, C is self-dual as $n = 2k$.

Week 6

The Shannon limit (optional material, not assessed)

Version 2023-10-17. To accessible online version of this chapter

Synopsis. *This chapter is under development.*

What is the main goal of Coding Theory?

Let codewords of a code C be transmitted via a given channel. Recall that the **probability of correct decoding** $P_{\text{corr}}(C)$ is the probability that a codeword picked at random from C and sent via the channel is decoded correctly.

It is not difficult to see that, unless the channel cannot transmit any information (e.g., $BSC(0.5)$), a code can be chosen so that this probability is arbitrarily close to 1. For example, one can use repetition codes Rep(n, F) with large n. However, the rate of Rep(n, F) is $1/n$ which tends to 0 as $n \to \infty$.

Shannon proved that it is possible to do much better than that. The following theorem is not examinable:

Theorem 6.1: Shannon's noisy-channel coding theorem

Each channel has capacity Cap, $0 \leq Cap \leq 1$, such that:

- for any $R < Cap$, there exist a sequence of codes C_n of rate $\geq R$ so that $P_{\text{corr}}(C_n)$ tends to 1.
- For each $R' > Cap$, there exists a positive ϵ such that $P_{\text{corr}}(C) < 1 - \epsilon$ for all codes C of rate $\geq R$.

However, his was an *existence proof* not giving any constructive way to find the codes. One can create codes C_n by picking codewords from F^n at random, but encoding and decoding for such codes are computationally unfeasible because random codes lack structure.

So a specific goal of Coding Theory is to **construct** sequences of codes which approach *Shannon's limit* — at least binary codes for $BSC(p)$, the capacity of which was calculated by Shannon to be $Cap(BSC(p)) = 1 - p\log_2 p^{-1} - (1-p)\log_2(1-p)^{-1}$. Note that the capacity of $BSC(0.5)$ is 0 (this channel outputs random junk irrespective of the input) but the capacity of $BSC(p)$ is positive if $p < 0.5$.

In fact, constructible codes which come close to Shannon's limit were only developed at the end of last century, and became part of Ethernet, Wi-Fi etc within the last decade. It took about fifty years to build up mathematical apparatus to achieve this.

Week 6

Exercises (answers at end)

Version 2023-10-17. To accessible online version of these exercises

Exercise 6.1 (optional, not discussed in class)**.** Show that some standard families of binary codes do not have rates approaching capacity of $BSC(p)$.

Week 6

Exercises — solutions

Version 2023-10-17. To accessible online version of these exercises

Exercise 6.1 (optional, not discussed in class)**.** Show that some standard families of binary codes do not have rates approaching capacity of $BSC(p)$.

Answer to E6.1.

Week 7

Hamming codes

Version 2023-11-04. To accessible online version of this chapter

Synopsis. *Hamming codes are essentially the first non-trivial family of codes that we shall meet. We give a construction of a q-ary Hamming code and prove that it is perfect with minimum distance 3. We show that syndrome decoding works for Hamming codes in an especially simple way.*

Finding a check matrix

Before we can construct Hamming codes, we need to discuss check matrices further and prove a result (the Distance Theorem) which will allow us to find the minimum distance of a linear code from its check matrix.

The following result allows us to find a generator matrix of $C^\perp$, assuming that C has a generator matrix in standard form.

Theorem 7.1: a check matrix construction

Assume that C has a $k \times n$ generator matrix $G = [\, I_k \,|\, A \,]$ in standard form. Then the dual code $C^\perp$ has generator matrix

$$H = [\, -A^T \,|\, I_{n-k} \,].$$

Proof. H has $n-k$ rows which are linearly independent (due to I_{n-k} present). It is enough to show that each row $\underline{r}$ of H is a codevector of $C^\perp$: indeed, we have $n-k$ linearly independent vectors in $C^\perp$, and $n-k$ is the dimension of $C^\perp$ by Theorem 5.1, so a linearly independent set of $n-k$ vectors must be a basis of $C^\perp$.

By Theorem 5.1, it is enough to show that $\underline{r}G^T = \underline{0}$. We will show this at once for all rows of H, by proving that HG^T is the zero matrix. Indeed,

$$[\,-A^T \,|\, I_{n-k}\,] \left[\frac{I_k}{A^T} \right] = -A^T I_k + I_{n-k} A^T = -A^T + A^T = 0. \qquad \square$$

How can one find a check matrix of C if C has no generator matrix in standard form? We address this question below.

Linearly equivalent codes

Definition: linearly equivalent codes

Two linear codes $C, C' \subseteq \mathbb{F}_q^n$ are **linearly equivalent**, if C' can be obtained from C by a sequence of linear transformations of the following types:

(C1) choose indices i, j; in every codeword, swap symbols x_i and x_j;

(C2) choose index i and non-zero $\lambda \in \mathbb{F}_q$; in every codeword, multiply x_i by λ.

Exercise. Linearly equivalent codes have the same length, dimension and weight. They have the same weight enumerator. (*Reason*: (C1) and (C2) do not change the weight of any vector.)

Fact: known from linear algebra

Every generator matrix can be brought into the standard form by using row operations (R1), (R2), (R3) considered above and column operations (C1).

Reason: any matrix can be brought to reduced row echelon form, RREF, by (R1)–(R3); a generator matrix has linearly independent rows so the RREF won't have zero rows and will have a leading entry 1 in each of the k rows; the k columns which contain the leading entries are columns of the identity matrix of size k; use (C1) to move all these columns to the left.

Conclusion: we can always find a generator matrix in standard form for a linearly equivalent code.

The Distance Theorem

We already know how to read the length and the dimension of a linear code C off a check matrix H of C:

- the number of columns of H is the length of C;
- the number of columns minus the number of rows of H is the dimension of C.

The following theorem tells us how to determine the minimum distance of C using H.

Theorem 7.2: Distance Theorem for linear codes

Let $C \subseteq \mathbb{F}_q^n$ be a linear code with check matrix H. Then $d(C) = d$ if and only if every set of $d-1$ columns of H is linearly independent and some set of d columns of H is linearly dependent.

Proof. Let e be the size of a smallest linearly dependent subset of the set $\{\overline{h}_1, \ldots, \overline{h}_n\}$ of columns of H. The theorem claims that $e = d(C)$. Note that e is the minimum positive number of non-zero coefficients x_i in the linear combination

$$x_1\overline{h}_1 + x_2\overline{h}_2 + \ldots + x_n\overline{h}_n = \overline{0},$$

i.e., the minimum weight of non-zero $\underline{x} = (x_1, \ldots, x_n)$ such that $\underline{x}H^T = \underline{0}$. By Theorem 5.1, such vectors $\underline{x}$ are exactly the codevectors of C, so $e = w(C) = d(C)$ as claimed. $\square$

Example: calculate $d(C)$ using the Distance Theorem

Use the Distance Theorem to find the minimum distance of the ternary linear code with check matrix $H = \begin{bmatrix} 0 & 1 & 2 & 1 \\ 2 & 0 & 1 & 1 \end{bmatrix}$.

Solution. *Step 1.* H has no zero columns. Hence every set of 1 column is linearly independent (a one-element set is linearly dependent iff that element is zero). So $d \geq 2$.

Step 2. Any two columns of H are linearly independent, because no two columns are proportional to each other. So $d \geq 3$.

Step 3. There are three linearly dependent columns in H: for example, columns 1, 2 and 3 form linear combination $\begin{bmatrix} 0 \\ 2 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} 2 \\ 1 \end{bmatrix} = \overline{0}$. Therefore, $d = 3$.

Hamming codes: the construction

Definition: line, representative vector, projective space

A **line** is a 1-dimensional subspace of the vector space $\mathbb{F}_q^n$.
A **representative vector** of a line is a non-zero vector $\underline{u}$ from that line. The line is then given by $\{\lambda\underline{u} \mid \lambda \in \mathbb{F}_q\}$.
The **projective space** $\mathbb{P}_{n-1}(\mathbb{F}_q)$ is the set of all lines in $\mathbb{F}_q^n$.

Remark: the terminology comes from euclidean geometry — in the euclidean plane, the set of all vectors proportional to a given non-zero vector is a straight line through the origin. Projective spaces over the field $\mathbb{R}$ of real numbers are well-studied geometric objects.

For example, $\mathbb{P}_1(\mathbb{R})$ — the set of all lines through the origin in the euclidean plane — can be thought of as the unit circle with antipodes identified. We are working over a finite field $\mathbb{F}_q$ where these notions are less intuitive.

Definition: Hamming codes

Let $r \geq 2$ be given. We let $\mathbf{Ham}(\boldsymbol{r}, \boldsymbol{q})$ denote an $\mathbb{F}_q$-linear code whose check matrix has columns which are representatives of the lines in $P_{r-1}(\mathbb{F}_q)$, exactly one representative vector from each line.

Remark: $\mathrm{Ham}(r, q)$ **is not one code but a class of linearly equivalent codes**

$\mathrm{Ham}(r, q)$ is defined up to a linear equivalence. Indeed, we can:

- multiply a column by non-zero λ to get another representative of the same line;
- put columns in any order.

This means that $\mathrm{Ham}(r, q)$ is not just one code but a class of linearly equivalent codes. We will therefore say "*a* $\mathrm{Ham}(r, q)$ *code*" to mean any of the linearly equivalent codes.

Let us see how the construction works in historically the first example of a Hamming code.

Example: $\mathrm{Ham}(3, 2)$

Construct a parity check matrix for a binary Hamming code $\mathrm{Ham}(3, 2)$. Then find a generator matrix in standard form for $\mathrm{Ham}(3, 2)$.

Solution: we need to take one non-zero column from each line in $\mathbb{F}_2^3$. For binary vectors, a line $\{\lambda \underline{u} \mid \lambda \in \mathbb{F}_2\}$ consists only of two points, $\underline{0}$ and $\underline{u}$. This means that a check matrix for a **binary** Hamming code consists of **all non-zero binary columns** or the required size.

Start filling in the check matrix by putting the identity columns at the end (this is convenient for finding a generator matrix). In total, there are 7 non-zero binary vectors of size 3:

$$H = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{array}\right]$$

From this H, we can reconstruct the generator matrix $G = [I_k \mid A]$ by Theorem 7.1:

$$G = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

This is, up to linear equivalence, the generator matrix of the original code of R. Hamming.

Historical remark. Despite their name, the q-ary Hamming codes for $q > 2$ were not invented by Hamming. Richard Hamming told Claude Shannon (who he shared an office with at Bell Labs) about his binary $[7,4,3]$-code, and Shannon mentioned it in his paper of 1948. That paper was read by **Marcel J. E. Golay** (1902–1989), a Swiss-born American mathematician and electronics engineer, who then suggested the $\mathrm{Ham}(r,q)$ construction in his paper published in 1949. Golay went further and constructed two perfect codes which are not Hamming codes. He asked whether there are any more perfect codes.

We will see the Golay codes, and will learn about an answer to Golay's question about perfect codes, later in the course.

Parameters of a Hamming code

We considered an example of a $\mathrm{Ham}(3,2)$ code, which — by looking at its generator matrix — turns out to be a $[7,4,d]_2$ code. It is not difficult to see directly that $d = 3$. By explicitly computing the Hamming bound, one can show that all $[7,4,3]_2$-codes are perfect.

We will now generalise this and show that all Hamming codes are perfect.

Theorem 7.3: properties of Hamming codes

$\mathrm{Ham}(r,q)$ is a perfect $[n,k,d]_q$ code where $n = \dfrac{q^r - 1}{q-1}$, $k = n - r$, $d = 3$.

Proof. The length n of the code is equal to the number of columns in the check matrix, which is $\#\mathbb{P}_{r-1}(\mathbb{F}_q)$, the number of lines in $\mathbb{F}_q^r$.

Observe that two lines intersect only at one point, namely $\overline{0}$. The set $\mathbb{F}_q^r \setminus \{\overline{0}\}$ is therefore a disjoint union of lines. Each line $\{\lambda\overline{u} : \lambda \in F\}$ contains $q-1$ non-zero points.

So the number of lines in $\mathbb{F}_q^r$ can be found as $\dfrac{\#(\mathbb{F}_q^r \setminus \{\overline{0}\})}{q-1} = \dfrac{q^r-1}{q-1}$.

We have $k = \dim \mathrm{Ham}(r,q) = n - r$ since, by construction, the check matrix H has r rows.

To find d, we use the Distance Theorem for linear codes. Any two columns of H are linearly independent because they are from different lines in $\mathbb{F}_q^r$. (Two vectors are linearly dependent only if they are proportional to each other, i.e., belong to the same line.) Therefore, $d \geq 3$.

On the other hand, H has columns $(a, 0, 0, \ldots, 0)^T$, $(0, b, 0, \ldots, 0)^T$ and $(c, c, 0, \ldots, 0)^T$, from three different lines (where $a, b, c \in \mathbb{F}_q \setminus \{0\}$). These columns are linearly dependent:

$$a^{-1}\begin{bmatrix} a \\ 0 \\ \vdots \\ 0 \end{bmatrix} + b^{-1}\begin{bmatrix} 0 \\ b \\ \vdots \\ 0 \end{bmatrix} - c^{-1}\begin{bmatrix} c \\ c \\ \vdots \\ 0 \end{bmatrix} = \overline{0}.$$

So $d = 3$ by the Distance Theorem.

It remains to show that $\mathrm{Ham}(r, q)$ is perfect. We calculate $t = [(d-1)/2] = [2/2] = 1$. The Hamming bound (in logarithmic form) then says

$$k \le n - \log_q\left(\binom{n}{0} + \binom{n}{1}(q-1)\right) = n - \log_q\left(1 + n(q-1)\right).$$

By the already proved formulae for n and k we have $n(q-1) = q^r - 1$ and $k = n - r$. Hence the bound is $n - r \le n - \log_q(q^r) = n - r$ — attained. Thus, $\mathrm{Ham}(r, q)$ is perfect. $\square$

Remark: $(q^r - 1)/(q - 1)$ is an integer

The proof shows that the fraction $\dfrac{q^r - 1}{q - 1}$ is an integer. In fact, this can be seen for all integers $q, r > 1$ by a formula for summing a geometric progression, $\dfrac{q^r - 1}{q - 1} = q^{r-1} + q^{r-2} + \cdots + q + 1$; the right-hand side is obviously an integer.

Decoding a Hamming code

Algorithm 7.4: decoding algorithm for a Hamming code

Let a Hamming code be given by its check matrix H. Suppose a vector $\underline{y}$ is received.

- Calculate $S(\underline{y}) = \underline{y}H^T$. If $S(\underline{y}) = \underline{0}$, $\texttt{DECODE}(\underline{y}) = \underline{y}$.
- Otherwise, $S(\underline{y}) = \lambda \times$ some column of H. Let this be the ith column of H.
- Subtract λ from the ith position in $\underline{y}$. The result is the codevector $\texttt{DECODE}(\underline{y})$.

Proof of validity of the algorithm. We prove that the algorithm outputs the nearest neighbour of $\underline{y}$ in the code C. This is clear if $S(\underline{y}) = \underline{y}H^T = \underline{0}$: by Proposition 5.2 $\underline{y}$ is a codevector, and so its own nearest neighbour in C. Hence it is correct to decode $\underline{y}$ to itself.

If $\underline{y}H^T \neq \underline{0}$, the line in $\mathbb{F}_q^r$ which contains $\underline{y}H^T$ has a representative column in H — say, h_i. As $\underline{y}H^T$ lies on the line spanned by h_i, we must have $\underline{y}H^T = \lambda h_i$ for some $\lambda \in \mathbb{F}_q$.

Note that λh_i equals $(\lambda \underline{e}_i)H^T$ where $\underline{e}_i$ is the unit vector with symbol 1 in position i and zeros elsewhere. It follows that

$$(\underline{y} - \lambda \underline{e}_i)H^T = \lambda h_i - \lambda h_i = \underline{0},$$

hence $\underline{y} - \lambda \underline{e}_i$ is a codevector. Finally, since $d(\underline{y}, \underline{y} - \lambda \underline{e}_i) = 1$, and no codevector can be at distance **less** than 1 from $\underline{y}$, we conclude that $\underline{y} - \lambda \underline{e}_i$ is the nearest neighbour of $\underline{y}$ in C. $\square$

Remark: properties of a Hamming decoder

The Algorithm and the proof above imply:

- Every coset leader of $\mathrm{Ham}(r, q)$ is $\underline{0}$ or $\lambda \underline{e}_i$, i.e., a vector of weight 0 or 1.
- The decoder changes at most one symbol in the received vector.

Note that the fact that every coset leader is of weight ≤ 1 also follows, in a different way, from Exercise 4.3.

For $\mathrm{Ham}(3, 2)$, a clever ordering of columns in the parity check matrix can make the decoding algorithm especially elegant:

Example: special check matrix for $\mathrm{Ham}(3, 2)$

Construct a decoder for the $\mathrm{Ham}(3, 2)$ code with parity check matrix

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

Solution. If $\underline{y} \in \mathbb{F}_2^7$ is received, $\underline{y}H^T$ is either $\underline{0}$ or one of the columns of H. Now note that, by Algorithm 7.4,

- if $\underline{y}H^T = 001$, the decoder must subtract 1 from the first bit in $\underline{y}$, because 001 is the first column of H;
- if $\underline{y}H^T = 010$, the decoder must subtract 1 from the second bit in $\underline{y}$, because 010 is the second column of H;

and so on. Subtracting 1 from a bit in $\mathbb{F}_2$ is the same as "flipping" the bit, i.e., replacing 0 by 1 and 1 by 0.

Thus, to decode the received vector $\underline{y}$, we calculate the syndrome $\underline{y}H^T$. If this is 000, output $\underline{y}$, otherwise *read the syndrome $\underline{y}H^T$ as the binary representation of a number $i \in \{1, 2, \ldots, 7\}$ and decode by flipping the ith bit in $\underline{y}$.*

Week 7

Exercises (answers at end)

Version 2023-11-07. To accessible online version of these exercises

Notation: let $\mathcal{H}_7$ denote a $\mathrm{Ham}(3,2)$ code. It is a $[7,4,3]_2$ linear code.

Exercise 7.1. Construct a generator matrix for $\mathcal{H}_7$. Hence write down all the codevectors of $\mathcal{H}_7$ and find the weight enumerator $W_{\mathcal{H}_7}(x,y)$.

Exercise 7.2. (a) If $\underline{v} = (x_1, x_2, \ldots, x_n)$ is a binary vector, we extend $\underline{v}$ to obtain the vector $\widehat{\underline{v}} = (x_1, \ldots, x_n, x_{n+1})$ where $x_{n+1} = x_1 + \ldots + x_n$ in $\mathbb{F}_2$. That is, a vector is extended by appending one bit so that the resulting vector has even weight.

If C is a binary linear code, we define the **extended code** C as $\{\widehat{\underline{c}} : \underline{c} \in C\}$. The extended $\mathcal{H}_7$ is denoted $\mathcal{H}_8$.

(a) By looking at $W_{\mathcal{H}_7}$, show that the weight enumerator of the extended Hamming code is $W_{\mathcal{H}_8}(x,y) = x^8 + 14x^4y^4 + y^8$. Determine the length, dimension and weight of $\mathcal{H}_8$, state how many bit errors per codeword can $\mathcal{H}_8$ detect and correct.

(b) Show: if $\underline{u}$, $\underline{v} \in \mathbb{F}_2^n$ are such that $w(\underline{u}), w(\underline{v}), w(\underline{u}+\underline{v})$ are divisible by 4, then $\underline{u} \cdot \underline{v} = 0$.

(c) Deduce from (a) and (b) that $\mathcal{H}_8$ is a self-dual code.

(d) Write down a generator matrix $\widehat{G}$ for $\mathcal{H}_8$ and use it to prove directly that $\mathcal{H}_8$ is self-dual.

Week 7

Exercises — solutions

Version 2023-11-07. To accessible online version of these exercises

Notation: let $\mathcal{H}_7$ denote a $\mathrm{Ham}(3,2)$ code. It is a $[7,4,3]_2$ linear code.

Exercise 7.1. Construct a generator matrix for $\mathcal{H}_7$. Hence write down all the codevectors of $\mathcal{H}_7$ and find the weight enumerator $W_{\mathcal{H}_7}(x,y)$.

Answer to E7.1. Let us use the following parity check matrix: $H = \begin{bmatrix} 1&1&1&0&1&0&0\\ 1&1&0&1&0&1&0\\ 1&0&1&1&0&0&1 \end{bmatrix}$.

(Any other check matrix for a $\mathrm{Ham}(3,2)$ code is obtained from this one by permuting columns.) This allows us to construct the corresponding generator matrix for $\mathcal{H}_7$:

$G = \begin{bmatrix} 1&0&0&0&1&1&1\\ 0&1&0&0&1&1&0\\ 0&0&1&0&1&0&1\\ 0&0&0&1&0&1&1 \end{bmatrix}$, hence the code consists of $[0000]G = 0000000$;

$$\begin{aligned}
&[0001]G = 0001011,\ [0010]G = 0010101,\ [0011]G = 0011110,\\
&[0100]G = 0100110,\ [0101]G = 0101101,\ [0110]G = 0110011,\ [0111]G = 0111000,\\
&[1000]G = 1000111,\ [1001]G = 1001100,\ [1010]G = 1010010,\ [1011]G = 1011001,\\
&[1100]G = 1100001,\ [1101]G = 1101010,\ [1110]G = 1110100,
\end{aligned}$$

$[1111]G = 1111111$. One codevector has weight 0, seven have weight 3, seven have weight 4 and one has weight 7. Hence the weight enumerator is

$$W_{\mathcal{H}_7}(x,y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7.$$

Exercise 7.2. (a) If $\underline{v} = (x_1, x_2, \dots, x_n)$ is a binary vector, we extend $\underline{v}$ to obtain the vector $\widehat{\underline{v}} = (x_1, \dots, x_n, x_{n+1})$ where $x_{n+1} = x_1 + \dots + x_n$ in $\mathbb{F}_2$. That is, a vector is extended by appending one bit so that the resulting vector has even weight.

If C is a binary linear code, we define the **extended code** C as $\{\widehat{\underline{c}} : \underline{c} \in C\}$. The extended $\mathcal{H}_7$ is denoted $\mathcal{H}_8$.

(a) By looking at $W_{\mathcal{H}_7}$, show that the weight enumerator of the extended Hamming code is $W_{\mathcal{H}_8}(x, y) = x^8 + 14x^4y^4 + y^8$. Determine the length, dimension and weight of $\mathcal{H}_8$, state how many bit errors per codeword can $\mathcal{H}_8$ detect and correct.

(b) Show: if $\underline{u}$, $\underline{v} \in \mathbb{F}_2^n$ are such that $w(\underline{u}), w(\underline{v}), w(\underline{u} + \underline{v})$ are divisible by 4, then $\underline{u} \cdot \underline{v} = 0$.

(c) Deduce from (a) and (b) that $\mathcal{H}_8$ is a self-dual code.

(d) Write down a generator matrix $\widehat{G}$ for $\mathcal{H}_8$ and use it to prove directly that $\mathcal{H}_8$ is self-dual.

Answer to E7.2. (a) By the extended code construction, $\mathcal{H}_8$ has length 8 and cardinality equal to $\#\mathcal{H}_7$, hence dimension 4.

If $\underline{x} \in \mathcal{H}$ is a codevector of weight 3 or 4 (there are 14 such vectors in $\mathcal{H}$), $\widehat{\underline{x}}$ will be of weight 4. Hence $\mathcal{H}_8$ contains $\underline{0}$, fourteen vectors of weight 4 and 11111111 of weight 8. So the weight enumerator is as claimed.

(b) Call the number of positions i where $u_i = v_i = 1$ the **overlap** of $\underline{u}$ and $\underline{v}$. Note that in $\underline{u} + \underline{v}$, the number of 1s is exactly $w(\underline{u}) + w(\underline{v}) - 2 \times$ overlap of $\underline{u}$ and $\underline{v}$. The weight of $\underline{u} + \underline{v}$ must be divisible by 4, so the overlap is even. But then, $\underline{u} \cdot \underline{v}$ is the sum of an even number of 1s, so $\underline{u} \cdot \underline{v} = 0$.

(c) If $\underline{x}$ and $\underline{y}$ are codevectors of $\mathcal{H}_8$, then $\underline{x} + \underline{y}$ is also a codevector of $\mathcal{H}_8$, and (a) tells us that the weights of all codevectors are divisible by 4. Hence by (b) $\underline{x} \cdot \underline{y} = 0$. This means that $\mathcal{H}_8 \subseteq \mathcal{H}_8^{\perp}$. The dimension of both codes is 4, hence they are equal.

(d) Appending a parity check bit to each row of G above, we obtain a generator matrix $\widehat{G}$ for $\mathcal{H}_8$: $\widehat{G} = \left[\begin{array}{ccccccc|c} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$. A direct calculation shows that $\widehat{G}\widehat{G}^T$ is zero. So $\mathcal{H}_8$ is self-orthogonal and, given that $2k = n$ $(2 \times 4 = 8)$, is self-dual.

Week 8

The MacWilliams identity. The Average Weight Equation. Plotkin bound. Simplex codes

Version 2023-11-07. To accessible online version of this chapter

Synopsis. *Remarkably, the weights of codevectors of the dual code $C^{\perp}$ are completely determined by weights of codevectors of C. This was proved by* **Florence Jessie MacWilliams** *(1917–1990), an English-born American mathematician who spent most of her career at Bell Labs and Harvard in the United States. We state the general case of the MacWilliams identity. We give a proof (not examinable) for codes over $\mathbb{F}_p$ with prime p, and apply the identity to deduce a formula called the Average Weight Equation, as well as the Plotkin bound. We can use the MacWilliams identity to study Hamming codes by analysing their dual codes, called* **simplex codes.**

Theorem 8.1: the MacWilliams identity

If C is a q-ary linear code, $W_{C^{\perp}}(x, y) = \dfrac{1}{\#C} W_C(x + (q-1)y,\, x - y)$.

Proof for prime $q = p$. This proof is not examinable. Since p is a prime, the field $\mathbb{F}_p$ consists of elements $0, 1, \dots, p-1$ (residues of integers modulo p). Being able to explicitly list the field elements — not possible for a general prime power q — simplifies the proof.

Let $C \subseteq \mathbb{F}_p^n$ be linear. We fix the complex number $\omega = e^{2\pi i/p}$, a primitive pth root of 1. We have $\omega^p = 1$ and $\omega, \omega^2, \dots, \omega^{p-1} \neq 1$. We can write ω^a if $a \in \mathbb{F}_p$ — this complex number is well-defined, even though a is only defined modulo p.

Given $\underline{c} \in C$, $\underline{v} \in \mathbb{F}_p^n$, denote

$$\Phi(\underline{c}, \underline{v}) = \omega^{\underline{c}\cdot\underline{v}} x^{n-w(\underline{v})} y^{w(\underline{v})}.$$

We will compute $\sum_{\underline{c}\in C,\ \underline{v}\in\mathbb{F}_p^n} \Phi(\underline{c},\underline{v})$ in two different ways.

Way 1. If $\underline{v}\in C^\perp$, then $\underline{c}\cdot\underline{v}=0$ for all $\underline{c}\in C$, so $\Phi(\underline{c},\underline{v}) = x^{n-w(\underline{v})}y^{w(\underline{v})}$.

If, however, $\underline{v}\notin C^\perp$, there is a codevector $\underline{d}\in C$ such that $\underline{d}\cdot\underline{v}=a\neq 0$ in $\mathbb{F}_p$. Observe that $\Phi(\underline{d}+\underline{c},\underline{v}) = \omega^{\underline{d}\cdot\underline{v}}\Phi(\underline{c},\underline{v}) = \omega^a\Phi(\underline{c},\underline{v})$. We know that $\underline{d}+C=C$, so

$$\sum_{\underline{c}\in C}\Phi(\underline{c},\underline{v}) = \sum_{\underline{c}\in C}\Phi(\underline{d}+\underline{c},\underline{v}) = \omega^a\sum_{\underline{c}\in C}\Phi(\underline{c},\underline{v}) \qquad\Longrightarrow\qquad (\omega^a-1)\sum_{\underline{c}\in C}\Phi(\underline{c},\underline{v})=0.$$

Since $\omega^a\neq 1$, we have

$$\sum_{\underline{c}\in C}\Phi(\underline{c},\underline{v})=0 \quad \text{for } \underline{v}\notin C^\perp.$$

We conclude that

$$\sum_{\underline{c}\in C,\ \underline{v}\in\mathbb{F}_p^n}\Phi(\underline{c},\underline{v}) = \sum_{\underline{c}\in C,\ \underline{v}\in C^\perp}\Phi(\underline{c},\underline{v}) = \#C\sum_{\underline{v}\in C^\perp}x^{n-w(\underline{v})}y^{w(\underline{v})} = (\#C)W_{C^\perp}(x,y).$$

Way 2. If v is a symbol, $v\in\mathbb{F}_p$, we introduce the "weight of v", $w(v)$, as follows: $w(v)=1$ if $v\neq 0$ and $w(v)=0$ if $v=0$. Surely, for a vector $\underline{v}\in\mathbb{F}_p^n$ we have $w(\underline{v}) = w(v_1)+\cdots+w(v_n)$. We then rewrite

$$\begin{aligned}\Phi(\underline{c},\underline{v}) &= \omega^{c_1v_1+\cdots+c_nv_n}x^{1-w(v_1)}y^{w(v_1)}\ldots x^{1-w(v_n)}y^{w(v_n)}\\ &= \omega^{c_1v_1}x^{1-w(v_1)}y^{w(v_1)}\ldots\omega^{c_nv_n}x^{1-w(v_n)}y^{w(v_n)}.\end{aligned}$$

We now sum over $\underline{v}\in\mathbb{F}_p^n$ first: each coordinate of $\underline{v}$ runs over $\mathbb{F}_p=\{0,1,\ldots,p-1\}$. So, for a fixed $\underline{c}\in C$,

$$\begin{aligned}\sum_{\underline{v}\in\mathbb{F}_p^n}\Phi(\underline{c},\underline{v}) &= \sum_{v_1=0}^{p-1}\cdots\sum_{v_n=0}^{p-1}\Phi(\underline{c},\underline{v})\\ &= \sum_{v_1=0}^{p-1}\omega^{c_1v_1}x^{1-w(v_1)}y^{w(v_1)}\ldots\sum_{v_n=0}^{p-1}\omega^{c_nv_n}x^{1-w(v_n)}y^{w(v_n)}.\end{aligned} \tag{*}$$

Let us analyse the first factor in the product on the right-hand side of (*):

$$\sum_{v_1=0}^{p-1}\omega^{c_1v_1}x^{1-w(v_1)}y^{w(v_1)} = x + \big(\sum_{v_1=1}^{p-1}\omega^{c_1v_1}\big)y.$$

If $c_1=0$, the coefficient of y is clearly $1+1+\cdots+1=p-1$, whereas if $c_1\neq 0$, the coefficient of y is the sum of a geometric progression

$$\sum_{v_1=1}^{p-1}\omega^{c_1v_1} = -1+\sum_{v_1=0}^{p-1}\omega^{c_1v_1} = -1+\frac{1-(\omega^{c_1})^p}{1-\omega^{c_1}} = -1+\frac{0}{1-\omega^{c_1}} = -1$$

since $(\omega^{c_1})^p = 1$. Hence the first factor on the right-hand side of (*) is

$$\begin{cases} x+(p-1)y, & \text{if } c_1 = 0, \\ x-y, & \text{if } c_1 \neq 0. \end{cases}$$

The same applies to the second, ..., nth factor in (*), hence (*) has $w(\underline{c})$ factors equal to $x-y$ and $n-w(\underline{c})$ factors equal to $x+(p-1)y$. In other words, (*) evaluates as $(x+(p-1)y)^{n-w(\underline{c})}(x-y)^{w(\underline{c})}$. Therefore,

$$\sum_{\underline{c}\in C}\sum_{\underline{v}\in\mathbb{F}_p^n}\Phi(\underline{c},\underline{v}) = \sum_{\underline{c}\in C}(x+(p-1)y)^{n-w(\underline{c})}(x-y)^{w(\underline{c})} = W_C(x+(p-1)y,\, x-y).$$

Comparing *Way 2* and *Way 1*, we conclude that $W_C(x+(p-1)y,\, x-y) = (\#C)W_{C^\perp}(x,y)$. This is the MacWilliams identity for $q=p$. $\square$

Simple examples where the MacWilliams identity is used

Let us obtain a short formula for the weight enumerator of the trivial code $\mathbb{F}_q^n$ by writing $\mathbb{F}_q^n$ as the dual code of the **null code** $Null = \{\underline{0}\}$. Of course, every vector in $\mathbb{F}_q^n$ is orthogonal to $\underline{0}$ which explains why $\mathbb{F}_q^n = Null^\perp$.

Clearly, $\#Null = 1$ and $W_{Null}(x,y) = x^n$ because N has only one codevector, which is of weight 0. Now use the MacWilliams identity:

Example: the weight enumerator of the trivial code $\mathbb{F}_q^n$

$W_{\mathbb{F}_q^n}(x,y) = \frac{1}{\#Null}W_{Null}(x+(q-1)y, x-y) = (x+(q-1)y)^n.$

We can obtain the same formula for the weight enumerator of the trivial code $\mathbb{F}_q^n$ without the use of MacWilliams identity, see earlier exercises.

The binary ($q=2$) MacWilliams identity allows us to immediately obtain a short formula for the weight enumerator of the even weight code E_n. Indeed, $E_n = Rep(n,\mathbb{F}_2)^\perp$, and the binary repetition code has weight enumerator $W_{Rep(n,\mathbb{F}_2)}(x,y) = x^n + y^n$ (see example sheets). Also, $\#Rep(n,\mathbb{F}_2) = 2$. Hence

Example: the weight enumerator of E_n

$W_{E_n}(x,y) = \frac{1}{\#Rep(n,\mathbb{F}_2)}W_{Rep(n,\mathbb{F}_2)}(x+y, x-y) = \frac{1}{2}((x+y)^n + (x-y)^n).$

Using the binomial formula, we can expand this sum as $x^n + \binom{n}{2}x^{n-2}y^2 + \binom{n}{4}x^{n-4}y^4 + \ldots$ In particular, this proves that $w(E_n) = d(E_n) = 2$ as the lowest positive power of x in this polynomial is two.

The Average Weight Equation for linear codes

The proof of the following result involves a surprising use of the MacWilliams identity.

Theorem 8.2: the Average Weight Equation

If C is a q-ary linear code of length n, the average of the weights of all the codevectors of C is $(n-z)(1-q^{-1})$, where z is the number of zero columns in a generator matrix of C.

Proof. We count codevectors of weight 1 in the dual code $C^\perp$. By Theorem 5.1, $\underline{v} \in C^\perp$ iff $\underline{v}G^T = \underline{0}$ where G is a generator matrix of C. If $\underline{v}$ is of weight 1 with $v_i \neq 0$, then the ith column of G is zero. The non-zero v_i can be chosen in $q-1$ ways, so each zero column of G gives rise to $q-1$ vectors of weight 1 in $C^\perp$, and there are $z(q-1)$ such vectors in total.

We must get the same number as the coefficient of $x^{n-1}y$ in the weight enumerator $W_{C^\perp}(x,y)$, which by the MacWilliams identity equals

$$\frac{1}{\#C}W_C(x+(q-1)y, x-y) = \frac{1}{\#C}\sum_{\underline{v}\in C}(x+(q-1)y)^{n-w(\underline{v})}(x-y)^{w(\underline{v})}. \tag{8.1}$$

We put $x=1$ and work out the coefficient of y. By the Binomial Theorem,

$$\begin{aligned} (1+(q-1)y)^{n-w(\underline{v})} &= 1+(n-w(\underline{v}))(q-1)y &&+ \text{higher powers of } y, \\ (1-y)^{w(\underline{v})} &= 1-w(\underline{v})y &&+ \text{higher powers of } y, \end{aligned}$$

and so the coefficient of y in the product of these two expressions is

$$(n-w(\underline{v}))(q-1) - w(\underline{v}) = n(q-1) - qw(\underline{v}).$$

Summing over $\underline{v} \in C$ then dividing by $\#C$ gives the coefficient of y in (8.1) as $n(q-1) - q\frac{1}{\#C}\sum_{\underline{v}\in C} w(\underline{v})$. We thus get the equation

$$z(q-1) = n(q-1) - q\frac{1}{\#C}\sum_{\underline{v}\in C} w(\underline{v}),$$

hence the average of all weights, $\frac{1}{\#C}\sum_{\underline{v}\in C} w(\underline{v})$, is $(n-z)\frac{q-1}{q}$ as claimed. □

A simple example where we verify the Average Weight Equation

The easiest case where we can explicitly verify the Average Weight Equation is $C = Rep(n, \mathbb{F}_q)$, the q-ary repetition code of length n. The code consists of the zero vector and $q-1$ vectors of the form $aa\ldots a$ where $a \in \mathbb{F}_q \setminus \{0\}$, of weight n. The total number of codevectors is q. The one-row generator matrix $\begin{bmatrix}1 & 1 & \ldots & 1\end{bmatrix}$ of the code does not contain a zero column, so $z=0$. We arrive at the following

Example: average weight of a codevector of $\mathit{Rep}(n, \mathbb{F}_q)$

The average weight of a codevector of $\mathit{Rep}(n, \mathbb{F}_q)$ is

$$\frac{1 \times 0 + (q-1) \times n}{q} = n(1 - q^{-1}),$$

which agrees with the Average Weight Equation.

Exercise. Verify the Average Weight Equation by explicit calculation for the trivial code $\mathbb{F}_q^n$.

Simplex codes

What is the weight enumerator of $\mathrm{Ham}(r, q)$? This question can be answered using the MacWilliams identity. In the particular case $q = 2$, the answer can be explored further to give the probability P_{undetect} for the binary Hamming code (we do not pursue this here).

Recall from the previous chapter that the Hamming codes are defined via an interesting check matrix whose columns form *a maximal set of columns where no two columns are proportional*. What is the code *generated* by this matrix? We analyse these codes in the rest of this chapter.

Definition: simplex code

A **simplex code** $\Sigma(r, q)$ is defined as $\mathrm{Ham}(r, q)^{\perp}$.

Remark: recall that a *regular simplex* in an n-dimensional euclidean space $\mathbb{R}^n$ is a convex polytope whose vertices are $n+1$ points with the same distance between each pair of points. Thus, a 2-dimensional regular simplex is an equilateral triangle, and a 3-dimensional regular simplex is a regular tetrahedron. The following result motivates our terminology.

Theorem 8.3: properties of a simplex code

The simplex code $\Sigma(r, q)$ has length $n = (q^r - 1)/(q - 1)$ and dimension r. The Hamming distance between each pair of codevectors is q^{r-1}.

Proof. The length and dimension of $\Sigma(r, q) = \mathrm{Ham}(r, q)^{\perp}$ are dictated by the parameters of the Hamming code, see Theorem 7.3. It remains to calculate the distances.

Since $\Sigma(r, q)$ is linear, it suffices to show that every non-zero $\underline{v} \in \Sigma(r, q)$ has weight q^{r-1}.

By linear algebra, there is a basis of $\Sigma(r, q)$ which contains $\underline{v}$, hence $\underline{v}$ is the first row of some generator matrix H' of $\Sigma(r, q)$.

Since H' is a check matrix for $\mathrm{Ham}(r,q)$ and $d(\mathrm{Ham}(r,q)) = 3$, by Distance Theorem 7.2 no two columns of H' are proportional, hence the columns of H' represent distinct lines in $\mathbb{F}_q^r$. Therefore, the weight of $\underline{v}$ (the first row of H') is the number of lines where the *first* entry of a representative vector is not zero.

The total number of possible columns of size r with non-zero top entry is $(q-1)$ (choices for the top entry) $\times q^{r-1}$ (choices for the other entries which are unrestricted). But $(q-1)$ non-zero columns form a line, hence the number of required lines is $(q-1)q^{r-1}/(q-1) = q^{r-1}$. Hence $w(\underline{v}) = q^{r-1}$ as claimed. □

The weight enumerator of a binary Hamming code

By Theorem 8.3, the weight enumerator of the simplex code $\Sigma(r,q)$ is

$$W_{\Sigma(r,q)}(x,y) = x^n + (q^r - 1)x^{n-q^{r-1}}y^{q^{r-1}}$$

where $n = \dfrac{q^r - 1}{q-1}$. This formula reflects the fact that there is one codevector of weight 0 and $q^r - 1$ codevectors of weight q^{r-1} in $\Sigma(r,q)$.

The weight enumerator of $\mathrm{Ham}(r,q) = \Sigma(r,q)^\perp$ can then be obtained using the MacWilliams identity. We do this for a binary Hamming code.

Proposition 8.4: the weight enumerator of $\mathrm{Ham}(r,2)$

$W_{\mathrm{Ham}(r,2)}(x,y) = \frac{1}{n+1}\left((x+y)^n + n(x+y)^{\frac{n-1}{2}}(x-y)^{\frac{n+1}{2}}\right)$ where $n = 2^r - 1$.

Proof. The MacWilliams identity, Theorem 8.1, in the case of binary codes gives $W_{C^\perp}(x,y) = \frac{1}{\#C}W_C(x+y, x-y)$. We put $C = \Sigma(r,2)$ so that $C^\perp = \mathrm{Ham}(r,2)$. By Theorem 7.3, $n = 2^r - 1$ so that $\#C = 2^r = n+1$ and the weight of each non-zero codevector in $\Sigma(r,2)$ is $q^{r-1} = 2^{r-1} = \frac{n+1}{2}$. We also have $n - q^{r-1} = n - \frac{n+1}{2} = \frac{n-1}{2}$.

Substituting these in the MacWilliams identity, we obtain $W_{\mathrm{Ham}(r,2)}$ as stated. □

Example: weight enumerator of the “original” Hamming code

$$W_{\mathrm{Ham}(3,2)} = \frac{1}{8}\left((x+y)^7 + 7(x+y)^3(x-y)^4\right) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7.$$

Exercise: explicitly expand the left-hand side in the formula for $W_{\mathrm{Ham}(3,2)}$.

Exercise: Use Proposition 8.4 to show that every binary Hamming code contains the vector $111\ldots1$ (all bits equal to 1).

The Plotkin Bound

The Plotkin bound was obtained by Morris Plotkin in 1960 for arbitrary (not necessarily linear) binary codes. It applies to codes with very large minimum distance: $d > n/2$ where n is the length of the code. A proof of the general case of the bound by a direct counting argument can be found in the literature. We will only prove the statement for linear codes, which will serve as an example of the power of the MacWilliams identity and its corollary, the Average Weight Equation. (*Historical note*: the MacWilliams identity was proved in 1961, i.e., after the Plotkin bound.)

Proposition 8.5: The Plotkin bound for binary linear codes

If $C \subseteq \mathbb{F}_2^n$ is a linear code such that $d = d(C) > n/2$, then $\#C \leq \dfrac{d}{d - n/2}$.

Proof. Let $M = \#C$. The code C contains the zero vector, $\underline{0}$, and $M - 1$ vectors of weight *at least* d. Then the average weight of a codevector of C is *at least*

$$\frac{1 \times 0 + (M-1) \times d}{M} = (1 - \frac{1}{M})d.$$

So from the Average Weight Equation (where z is the number of zero columns in a generator matrix of C) we obtain

$$(n-z)(1 - \frac{1}{2}) \geq (1 - \frac{1}{M})d \quad \Longrightarrow \quad \frac{n}{2} \geq (1 - \frac{1}{M})d \quad \Longleftrightarrow \quad \frac{n}{2d} \geq 1 - \frac{1}{M}$$

so that $1/M \geq 1 - n/(2d) = (2d - n)/(2d)$ and $M \leq 2d/(2d - n)$, as claimed. $\square$

Week 8

Exercises (answers at end)

Version 2023-11-09. To accessible online version of these exercises

Notation: $\mathcal{H}_7$ denotes a $\mathrm{Ham}(3,2)$ code, $W_{\mathcal{H}_7}(x,y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7$.

Exercise 8.1. If $\underline{v} = (x_1, x_2, \ldots, x_n)$ is a binary vector, we extend $\underline{v}$ to obtain the vector $\widehat{\underline{v}}$ $= (x_1, \ldots, x_n, x_{n+1})$ where $x_{n+1} = x_1 + \ldots + x_n$ in $\mathbb{F}_2$. That is, a vector is extended by appending one bit so that the resulting vector has even weight.

If C is a binary linear code, we define the **extended code** C as $\{\widehat{\underline{c}} : \underline{c} \in C\}$. The extended $\mathcal{H}_7$ is denoted $\mathcal{H}_8$.

(a) By looking at $W_{\mathcal{H}_7}$, show that the weight enumerator of the extended Hamming code is $W_{\mathcal{H}_8}(x,y) = x^8 + 14x^4y^4 + y^8$. Determine the length, dimension and weight of $\mathcal{H}_8$, state how many bit errors per codeword can $\mathcal{H}_8$ detect and correct.

(b) Show: if $\underline{u}$, $\underline{v} \in \mathbb{F}_2^n$ are such that $w(\underline{u}), w(\underline{v}), w(\underline{u}+\underline{v})$ are divisible by 4, then $\underline{u} \cdot \underline{v} = 0$.

(c) Deduce from (a) and (b) that $\mathcal{H}_8$ is a self-dual code.

(d) Write down a generator matrix $\widehat{G}$ for $\mathcal{H}_8$ and use it to prove directly that $\mathcal{H}_8$ is self-dual.

Exercise 8.2. Recall: $\mathcal{H}_8$ is self-dual with weight enumerator $W_{\mathcal{H}_8}(x,y) = x^8 + 14x^4y^4 + y^8$.

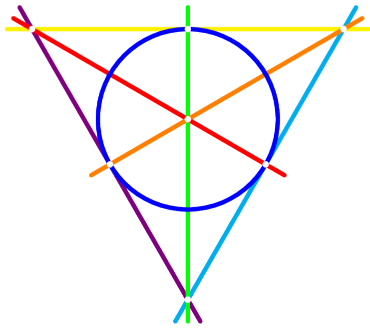
(a) Let $A(x,y) = x(x+y)$, $B(x,y) = (x-y)y$ and $C(x,y) = A - B = x^2 + y^2$. Show that $A(x+y, x-y) = 2A(x,y)$. Obtain similar equations for B and C.

(b) Show that $W_{\mathcal{H}_8} = C^4 - 4A^2B^2$.

(c) Deduce that $(\#\mathcal{H}_8)^{-1} W_{\mathcal{H}_8}(x+y, x-y) = W_{\mathcal{H}_8}(x,y)$.

(*Of course, this must be true by the MacWilliams identity, but the point of the exercise is to prove this algebraically.)*

Exercise 8.3 (a construction of the simplex code $\Sigma(3,2)$ using the Fano plane)**.** The **Fano plane** is the diagram of 7 points and 7 lines, given below. Each line passes through 3 points, and each points lies on 3 lines. For any two points $P \neq Q$, there exists exactly one line PQ which contains both P and Q.



Next to each point P, write a 7-bit word by the following rule. The bits are coloured red, orange, yellow, green, cyan, blue, violet. The red bit is **1** if P **does not** lie on the red line, and **0** if P lies on the red line. Same with the orange, ..., violet bit.

For example, the word written next to the point in the centre of the diagram above is **0010111**.

(a) Check that all the seven words are binary vectors of weight 4.

(b) Check that the seven words, together with the zero vector **0000000**, form a linear code in $\mathbb{F}_2^7$. To show that the code is closed under addition, prove that the sum of the words P and Q is the word R which is the third point on the line PQ.

(c) Prove that this linear code is $\Sigma(3, 2)$.

Week 8

Exercises — solutions

Version 2023-11-09. To accessible online version of these exercises

Notation: $\mathcal{H}_7$ denotes a $\mathrm{Ham}(3,2)$ code, $W_{\mathcal{H}_7}(x,y) = x^7 + 7x^4y^3 + 7x^3y^4 + y^7$.

Exercise 8.1. If $\underline{v} = (x_1, x_2, \ldots, x_n)$ is a binary vector, we extend $\underline{v}$ to obtain the vector $\widehat{\underline{v}}$ $= (x_1, \ldots, x_n, x_{n+1})$ where $x_{n+1} = x_1 + \ldots + x_n$ in $\mathbb{F}_2$. That is, a vector is extended by appending one bit so that the resulting vector has even weight.

If C is a binary linear code, we define the **extended code** C as $\{\widehat{\underline{c}} : \underline{c} \in C\}$. The extended $\mathcal{H}_7$ is denoted $\mathcal{H}_8$.

(a) By looking at $W_{\mathcal{H}_7}$, show that the weight enumerator of the extended Hamming code is $W_{\mathcal{H}_8}(x,y) = x^8 + 14x^4y^4 + y^8$. Determine the length, dimension and weight of $\mathcal{H}_8$, state how many bit errors per codeword can $\mathcal{H}_8$ detect and correct.

(b) Show: if $\underline{u}$, $\underline{v} \in \mathbb{F}_2^n$ are such that $w(\underline{u}), w(\underline{v}), w(\underline{u}+\underline{v})$ are divisible by 4, then $\underline{u} \cdot \underline{v} = 0$.

(c) Deduce from (a) and (b) that $\mathcal{H}_8$ is a self-dual code.

(d) Write down a generator matrix $\widehat{G}$ for $\mathcal{H}_8$ and use it to prove directly that $\mathcal{H}_8$ is self-dual.

Answer to E8.1. (a) By the extended code construction, $\mathcal{H}_8$ has length 8 and cardinality equal to $\#\mathcal{H}_7$, hence dimension 4.

If $\underline{x} \in \mathcal{H}_7$ is a codevector of weight 3 or 4 (there are 14 such vectors in $\mathcal{H}_7$), $\widehat{\underline{x}}$ will be of weight 4.

Indeed, if $\underline{x}$ is a vector of weight 3, then the extended vector $\widehat{\underline{x}}$ will be of weight 4. This is because 3 is odd and so the appended parity check bit will be 1, hence $w(\widehat{\underline{x}}) = w(\underline{x}) + 1$. If $\underline{x}$ is of weight 4 which is even, the appended parity check bit is 0 and so $w(\widehat{\underline{x}}) = w(\underline{x}) = 4$. Thus, the number of vectors of weight 4 in $\mathcal{H}_8$ is equal to the total number of vectors of weight 3 and weight 4 in $\mathcal{H}_7$.

Hence $\mathcal{H}_8$ contains $\underline{0}$, fourteen vectors of weight 4 and 11111111 of weight 8. So the weight enumerator is as claimed.

(b) Call the number of positions i where $u_i = v_i = 1$ the **overlap** of $\underline{u}$ and $\underline{v}$. Note that in $\underline{u} + \underline{v}$, the number of 1s is exactly $w(\underline{u}) + w(\underline{v}) - 2 \times$ overlap of $\underline{u}$ and $\underline{v}$. The weight of $\underline{u} + \underline{v}$ must be divisible by 4, so the overlap is even. But then, $\underline{u} \cdot \underline{v}$ is the sum of an even number of 1s, so $\underline{u} \cdot \underline{v} = 0$.

(c) If $\underline{x}$ and $\underline{y}$ are codevectors of $\mathcal{H}_8$, then $\underline{x} + \underline{y}$ is also a codevector of $\mathcal{H}_8$, and (a) tells us that the weights of all codevectors are divisible by 4. Hence by (b) $\underline{x} \cdot \underline{y} = 0$. This means that $\mathcal{H}_8 \subseteq \mathcal{H}_8^\perp$. The dimension of both codes is 4, hence they are equal.

(d) Appending a parity check bit to each row of G above, we obtain a generator matrix $\widehat{G}$ for $\mathcal{H}_8$: $\widehat{G} = \left[\begin{array}{ccccccc|c} 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{array}\right]$. A direct calculation shows that $\widehat{G}\widehat{G}^T$ is zero. So $\mathcal{H}_8$ is self-orthogonal and, given that $2k = n$ ($2 \times 4 = 8$), is self-dual.

Exercise 8.2. Recall: $\mathcal{H}_8$ is self-dual with weight enumerator $W_{\mathcal{H}_8}(x, y) = x^8 + 14x^4y^4 + y^8$.

(a) Let $A(x, y) = x(x + y)$, $B(x, y) = (x - y)y$ and $C(x, y) = A - B = x^2 + y^2$. Show that $A(x + y, x - y) = 2A(x, y)$. Obtain similar equations for B and C.

(b) Show that $W_{\mathcal{H}_8} = C^4 - 4A^2B^2$.

(c) Deduce that $(\#\mathcal{H}_8)^{-1} W_{\mathcal{H}_8}(x + y, x - y) = W_{\mathcal{H}_8}(x, y)$.

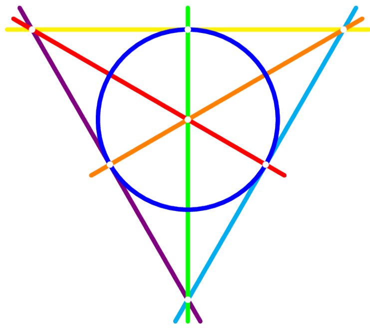
Answer to E8.2. (a) $A(x+y, x-y) = (x+y)(x+y+(x-y)) = 2(x+y)x = 2A(x, y)$. Similarly, $B(x + y, x - y) = 2B(x, y)$, same for C.

(b) Note that $AB = xy(x+y)(x-y) = xy(x^2 - y^2)$ so $4A^2B^2 = 4(x^6y^2 - 2x^4y^4 + x^2y^6)$. Subtracting this from $C^4 = x^8 + 4x^6y^2 + 6x^4y^4 + 4x^2y^6 + y^8$, we get $W_{\mathcal{H}_8}(x, y)$.

(c) The substitution of $(x+y, x-y)$ for (x, y) multiplies A, B and C by 2 hence multiplies $C^4 - 4A^2B^2$ by 2^4. Thus, $W_{\mathcal{H}_8}(x + y, x - y) = 16W_{\mathcal{H}_8}(x, y)$. Now note that $16 = \#\mathcal{H}_8$.

This approach is due to Andrew M. Gleason who supervised F.J. MacWilliams' PhD thesis.

Exercise 8.3 (a construction of the simplex code $\Sigma(3, 2)$ using the Fano plane)**.** The **Fano plane** is the diagram of 7 points and 7 lines, given below. Each line passes through 3 points, and each points lies on 3 lines. For any two points $P \neq Q$, there exists exactly one line PQ which contains both P and Q.



Next to each point P, write a 7-bit word by the following rule. The bits are coloured red, orange, yellow, green, cyan, blue, violet. The red bit is **1** if P **does not** lie on the red line, and **0** if P lies on the red line. Same with the orange, ..., violet bit.

For example, the word written next to the point in the centre of the diagram above is **0010111**.

(a) Check that all the seven words are binary vectors of weight 4.

(b) Check that the seven words, together with the zero vector **0000000**, form a linear code in $\mathbb{F}_2^7$. To show that the code is closed under addition, prove that the sum of the words P and Q is the word R which is the third point on the line PQ.

(c) Prove that this linear code is $\Sigma(3, 2)$.

Answer to E8.3.

(a) Each point lies on 3 lines, so each of the seven words will consist of 3 zeros and $7-3=4$ ones — thus be a vector in $\mathbb{F}_2^7$ of weight 4.

(b) Consider the words attached to points P, Q and R on the same line. We need to prove that $P+Q=R$, equivalently $P+Q+R=$ **0000000**.

Assume without the loss of generality that the line PQR is red. The red bits in P, Q and R are **0**; we have $\mathbf{0}+\mathbf{0}+\mathbf{0}=\mathbf{0}$. Consider any other bit colour, say blue. Exactly one of P, Q, R lies on the blue line, so for the blue bits we again have the correct equation $\mathbf{1}+\mathbf{1}+\mathbf{0}=\mathbf{0}$. Thus, $P+Q+R$ is indeed **0000000**.

(c) Pick three linearly independent vectors from the code, which means three points not on the same line: P, Q, R say. Use these three vectors as rows of the generator matrix.

I claim that the resulting matrix G with 3 rows will have 7 distinct non-zero columns.

First of all, no line contains P and Q and R, so there is no column $\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$ in G.

Secondly, the columns which correspond to the lines PQ, QR and PR will be $\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, $\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$ in some order. Indeed, each of these lines contains exactly two of the given points (hence two zeros in the column).

There are three further lines which contain exactly one of the P, Q, R, resulting in the columns $\left[\begin{smallmatrix}0\\1\\1\end{smallmatrix}\right]$, $\left[\begin{smallmatrix}1\\0\\1\end{smallmatrix}\right]$ and $\left[\begin{smallmatrix}1\\1\\0\end{smallmatrix}\right]$, and finally there is a line which contains none of P, Q, R, resulting in the column $\left[\begin{smallmatrix}1\\1\\1\end{smallmatrix}\right]$.

A matrix which consists of 7 pairwise distinct non-zero binary columns of size 3 is, by definition, a generator matrix for $\Sigma(3,2)$.

Explicitly, taking the vectors which correspond to the three points in the corners of the above diagram, we obtain the generator matrix

$$\begin{bmatrix} \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ \mathbf{1} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{0} \end{bmatrix},$$

which evidently has 7 distinct non-zero columns of size 3.

Week 9

Cyclic codes

Version 2023-11-12. To accessible online version of this chapter

Synopsis. *Cyclic codes form a subclass of linear codes. Cyclic codes are easy to define, but to reveal their advantages, one needs to study them using polynomials. We identify $\mathbb{F}_q^n$ with the space R_n of polynomials in $\mathbb{F}_q[x]$ of degree less than n, so that a linear code of length n becomes a subspace of R_n. We prove that cyclic codes are subspaces of very special form: a cyclic code C consists of all multiples, in R_n, of its generator polynomial $g(x)$. We also define a check polynomial of C. We can classify cyclic codes of length n by listing all monic divisors of the polynomial $x^n - 1$ in $\mathbb{F}_q[x]$. Theory and applications of cyclic codes are underpinned by the Division Theorem for polynomials and the long division algorithm, which we review here.*

Definition: cyclic shift, cyclic code

For a vector $\underline{a} = (a_0, a_1, \ldots, a_{n-1}) \in \mathbb{F}_q^n$, we denote $s(\underline{a}) = (a_{n-1}, a_0, \ldots, a_{n-2})$ and call the vector $s(\underline{a})$ the **cyclic shift** of $\underline{a}$.
A **cyclic code** in $\mathbb{F}_q^n$ is a **linear** code C such that $\forall \underline{a} \in C$, $s(\underline{a}) \in C$.
Equivalently, a cyclic code is a linear code C such that $s(C) = C$.

Remark: We can iterate the cyclic shift, so if a cyclic code C contains $(a_0, a_1, \ldots, a_{n-1})$, then C also contains the vectors $(a_{n-2}, a_{n-1}, a_0, \ldots, a_{n-3})$, $\ldots$, $(a_1, \ldots, a_{n-1}, a_0)$.

Vectors as polynomials

To study cyclic codes, we will identify **vectors of length n** with **polynomials of degree $< n$** with coefficients in the field $\mathbb{F}_q$:

$$\underline{a} = (a_0, a_1, \ldots, a_{n-1}) \quad \mapsto \quad a(x) = a_0 + a_1 x + \ldots + a_{n-1} x^{n-1} \quad \in \mathbb{F}_q[x]$$

Here $\mathbb{F}_q[x]$ is the **ring of polynomials** in one variable, x, with coefficients in $\mathbb{F}_q$.

Notation: the polynomial $a(x)$ and the vector $\underline{a}$

If n is given and $a(x)$ is a polynomial of degree less than n, $\underline{a}$ (same letter, underlined) will denote the vector which corresponds to $a(x)$ in $\mathbb{F}_q^n$.

Example: E_3 is a cyclic code

Show that the binary even weight code $E_3 = \{000, 110, 011, 101\} \subseteq \mathbb{F}_2^3$ is cyclic. List the **code polynomials** of E_3.

Solution. We know that E_3 is a linear code. It is closed under the cyclic shift: 000 is invariant under the cyclic shift, and $110 \xrightarrow{s} 011 \xrightarrow{s} 101$. Hence E_3 is a cyclic code:

Codevector	Code polynomial	Remark
000	0	
110	$1+x$	
011	$x+x^2$	$= x(1+x)$
101	$1+x^2$	$= (1+x)(1+x)$

We will soon explain the notable fact that all code polynomials of E_3 are multiples of $1+x$.

The Division Theorem for polynomials

In general we cannot divide $f(x)$ by $g(x)$ in $\mathbb{F}_q[x]$ and expect to get a polynomial. However, just as the ring $\mathbb{Z}$ of integers, the ring $\mathbb{F}_q[x]$ has an extra operation called **division with remainder**, as per the following

Theorem 9.1: Division Theorem for polynomials

For all $f(x) \in \mathbb{F}_q[x]$, $g(x) \in \mathbb{F}_q[x] \setminus \{0\}$, there exist unique $Q(x), r(x) \in \mathbb{F}_q[x]$ with

$$f(x) = g(x)Q(x) + r(x) \qquad \text{and} \qquad \deg r(x) < \deg g(x)$$

(possibly $r(x) = 0$). In this case the polynomial $Q(x)$ is the **quotient**, and $r(x)$ the **remainder**, of $f(x)$ when divided by $g(x)$.

We will **not** prove the Division Theorem but we will note and use the practical algorithm for finding the quotient and the remainder, known as **long division of polynomials**.

Example: long division of polynomials

Divide $x^5 + 1$ by $x^2 + x + 1$ in $\mathbb{F}_2[x]$, finding the quotient and the remainder.

Solution.

$$
\begin{array}{r|lll}
\multicolumn{1}{r}{} & x^3 + x^2 + 1 & & \text{(quotient)} \\ \cline{2-3}
x^2 + x + 1 & x^5 & +1 & \text{(dividend)} \\
 & -\,x^5 + x^4 + x^3 & & \\ \cline{2-3}
 & x^4 + x^3 & +1 & \\
 & -\,x^4 + x^3 + x^2 & & \\ \cline{2-3}
 & x^2 & +1 & \\
 & -\,x^2 + x + 1 & & \\ \cline{2-3}
 & x & & \text{(remainder)}
\end{array}
$$

Hence $x^5 + 1 = (x^2 + x + 1)Q(x) + r(x)$ in $\mathbb{F}_2[x]$, with $Q(x) = x^3 + x^2 + 1$ and $r(x) = x$.

This example shows long division of polynomials over $\mathbb{F}_2$. Division by a fixed binary polynomial is widely implemented in electronic circuits at hardware level, by means of shift feedback registers. We will soon see why such implementations are needed.

The generator polynomial of a cyclic code

In what follows, R_n denotes the space of polynomials of degree less than n.

Definition: generator polynomial

A **generator polynomial** of a cyclic code $C \subseteq R_n$, $C \neq \{0\}$ is a monic polynomial of least degree in C.
By convention, the generator polynomial of the null code $\{0\} \subseteq R_n$ is $x^n - 1$.

Recall that a polynomial $g(x)$ is **monic** if the coefficient of the highest power of x in $g(x)$ is 1.

Lemma 9.2: existence and uniqueness of a generator polynomial

Every cyclic code C has a unique generator polynomial $g(x)$.

Proof. If $C = \{0\}$, by definition $x^n - 1$ is the unique generator polynomial. Assume $C \neq \{0\}$.

Existence: take $g(x) \in C$ to be a non-zero polynomial of lowest degree in C. Make $g(x)$ monic by dividing it by its leading coefficient. This does not change the degree, so we now have a monic polynomial of least degree in C. Existence is proved.

Uniqueness: let $g_1(x) \in C$ be another generator polynomial, then by definition $g_1(x)$ is monic and has the same degree as $g(x)$. So $f(x) = g_1(x) - g(x)$ has degree less than $\deg g(x)$ (because the leading term $x^{\deg g}$ cancels due to subtraction). Note that $f(x) \in C$ because C is linear. If $f(x) \neq 0$, divide $f(x)$ by its leading coefficient and obtain a monic polynomial, again in C, of degree less than $\deg g$. This contradicts the choice of $g(x)$. Hence $f(x)$ must be 0, and $g_1(x) = g(x)$. Uniqueness is proved. $\square$

Theorem 9.3: properties of the generator polynomial

Let $C \subseteq R_n$ be a cyclic code with generator polynomial $g(x)$. Write $\deg g = n - k$. Then

1. $C = \{u(x)g(x) : u(x) \in R_k\}$, i.e., the code polynomials of C are all possible multiples of $g(x)$ of degree less than n.

2. $g(x)$ is a monic factor of the polynomial $x^n - 1$ in $\mathbb{F}_q[x]$.

Proof. Both claims are trivially true when $C = \{0\}$ and $g(x) = x^n - 1$, so assume $C \neq \{0\}$.

1. Observe that, writing elements of C as vectors, we have

$$\underline{g} = (g_0, g_1, \ldots, g_{n-k}, \underbrace{0, 0, \ldots, 0}_{k-1 \text{ zeros}})$$

and, as long as $i \leq k - 1$,

$$\underline{x^i g} = (\underbrace{0, \ldots, 0}_{i \text{ zeros}}, g_0, g_1, \ldots, g_{n-k}, \underbrace{0, \ldots, 0}_{k-1-i \text{ zeros}}).$$

That is, $\underline{x^i g}$ is obtained from $\underline{g}$ by applying the cyclic shift i times. Since C is cyclic, this means that $xg(x), \ldots, x^{k-1}g(x) \in C$.

Now, every polynomial $u(x) \in R_k$ — that is, a polynomial of degree less than k — is written as $u_0 + u_1 x + \cdots + u_{k-1}x^{k-1}$ for some $u_0, \ldots, u_{k-1} \in \mathbb{F}_q$. Hence $u(x)g(x)$ is a linear combination of the polynomials $g(x), xg(x), \ldots, x^{k-1}g(x)$ which are in C, and, as C is linear, $u(x)g(x) \in C$. We proved that $C \supseteq \{u(x)g(x) : u(x) \in R_k\}$.

Let us show that $C \subseteq \{u(x)g(x) : u(x) \in R_k\}$. Take $f(x) \in C$ and apply the Division Theorem for polynomials to write $r(x) = f(x) - g(x)Q(x)$ where $\deg r(x) < \deg g(x)$. We will get $\deg Q = \deg f - \deg g < n - (n - k) = k$ and so, by what has already been proved, $g(x)Q(x) \in C$. Then by linearity $r(x) \in C$. We have seen already that there cannot be a non-zero polynomial in C of degree strictly less than $\deg g$, so $r(x) = 0$ and $f(x) = g(x)Q(x)$ is a multiple of $g(x)$, as claimed. Part 1 of the Theorem is proved.

2. Continuing from the above, observe that

$$s(\underline{x^{k-1}g}) = (g_{n-k}, \underbrace{0, \ldots, 0}_{k-1 \text{ zeros}}, g_0, g_1, \ldots, g_{n-k-1})$$

where s is the cyclic shift. Hence the vector $s(\underline{x^{k-1}g})$ corresponds to the polynomial

$$g_{n-k} + x^k(g_0 + g_1x + \cdots + g_{n-k-1}x^{n-k-1})$$

which can be written as

$$g_{n-k} + x^k g(x) - g_{n-k}x^n = x^k g(x) - (x^n - 1),$$

as $g_{n-k} = 1$ given that $g(x)$ is monic. Since C is cyclic, $s(\underline{x^{k-1}g}) \in C$ and so $x^k g(x) - (x^n - 1) \in C$. Then by Part 1, $x^k g(x) - (x^n - 1) = u(x)g(x)$ for some polynomial $u(x)$, and so $x^n - 1 = (x^k - u(x))g(x)$ which shows that $g(x)$ is indeed a factor of $x^n - 1$. $\square$

Example: the generator polynomial of E_3

The code E_3 as a subspace of $\mathbb{F}_2[x]$ consists of polynomials 0, $1+x$, $x+x^2 = x(1+x)$ and $1 + x^2 = (1 + x)^2$. The generator polynomial of E_3 is $g(x) = 1 + x$ of degree 1.

As we have already noted, all the code polynomials of E_3 are multiples of $1 + x$.

Error detection by a cyclic code

Theorem 9.3 means that if C is a cyclic code, there is no need to store a check matrix for *error detection*. To determine whether the received vector $\underline{y}$ is a codevector, divide the polynomial $y(x)$ by the generator polynomial $g(x)$; the remainder is 0, if and only if $\underline{y} \in C$.

This is how error detection is implemented in practice for binary cyclic codes (e.g., in Ethernet networks). Long division by $g(x)$ is implemented by circuitry.

Nevertheless, for theoretical purposes we would like to have generator and check matrices for a cyclic code with a given generator polynomial.

The check polynomial

Definition: check polynomial

Let $g(x)$ be the generator polynomial of a cyclic code $C \subseteq \mathbb{F}_q^n$. The polynomial $h(x)$ defined by $g(x)h(x) = x^n - 1$ is the **check polynomial** of C.

Note that if $\deg g(x) = n - k$, then $\deg h(x) = k$, and h is monic.

Theorem 9.4: a generator matrix and a check matrix for a cyclic code

Let $C \subseteq \mathbb{F}_q^n$ be a cyclic code with generator polynomial $g(x) = g_0 + g_1 x + \ldots + g_{n-k}x^{n-k}$ and check polynomial $h(x) = h_0 + h_1 x + \ldots + h_k x^k$.
The vector $\underline{g}$ and its next $k-1$ cyclic shifts form a generator matrix for C:

$$G = \begin{bmatrix} g_0 & g_1 & \ldots & \ldots & g_{n-k} & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & \ldots & \ldots & g_{n-k} & \ddots & 0 \\ \vdots & \ddots & \ddots & & & & & \ddots \\ 0 & \ldots & 0 & g_0 & \ldots & \ldots & & g_{n-k} \end{bmatrix} \qquad (k \text{ rows}).$$

The vector of the polynomial

$$\overleftarrow{h}(x) = h_k + h_{k-1}x + \ldots + h_0 x^k,$$

obtained from $h(x)$ by reversing the order of the coefficients, and its next $n-k-1$ shifts form a check matrix for C:

$$H = \begin{bmatrix} 1 & h_{k-1} & \ldots & \ldots & h_1 & h_0 & 0 & \ldots & 0 \\ \vdots & \ddots & \ddots & & & & & \ddots & \\ 0 & \ddots & 1 & h_{k-1} & \ldots & \ldots & h_1 & h_0 & 0 \\ 0 & \ldots & 0 & 1 & \ldots & \ldots & & h_1 & h_0 \end{bmatrix} \qquad (n-k \text{ rows}).$$

Proof. The rows of G are linearly independent and the rows of H are linearly independent. Indeed, H is a matrix in a row echelon form with no zero rows, and so is G up to scaling of rows by a non-zero scalar g_0: note that $g_0 h_0 = g(0)h(0) = 0^n - 1 \neq 0$.

The linearly independent rows of G correspond to the polynomials $g(x), xg(x), \ldots, x^{k-1}g(x)$ and so they span $\{u(x)g(x) : \deg u(x) < k\}$ which by Theorem 9.3 is C. Thus, G is a generator matrix for C.

Since the number of rows of H is $n - k = \dim C^\perp$ and the rows are linearly independent, to show that H is a check matrix it is enough to show that $HG^T = 0$, same as in the proof of Theorem 7.1.

We express the inner product of vectors in terms of polynomials: if $\underline{a}, \underline{b} \in \mathbb{F}_q^n$, then

$$\underline{a} \cdot \underline{\overleftarrow{b}} = \text{coefficient of } x^{n-1} \text{ in } a(x)b(x).$$

Indeed, with $\underline{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\underline{\overleftarrow{b}} = (b_{n-1}, \ldots, b_1, b_0)$ one has $\underline{a} \cdot \underline{\overleftarrow{b}} = a_0 b_{n-1} + \cdots + a_{n-1}b_0$ which is exactly the coefficient of x^{n-1} in the product of the polynomials $a(x)$ and $b(x)$.

Number the rows of G from 0 to $k-1$, the rows of H from 0 to $n-k-1$. The rows of G are $\underline{x^i g}$, and the rows of H are the vectors of $x^j h$ written backwards. So an entry of HG^T, which as we know is an inner product of a row of G and a row of H, is the coefficient of x^{n-1} in $x^i g(x) x^j h(x) = x^{n+i+j} - x^{i+j}$. But since $n+i+j > n-1$ and $i+j < n-1$, this coefficient is zero, proving $HG^T = 0$. ☐

Remark: this is not the only generator matrix (resp., check matrix) for C. As we know, a generator matrix is not unique. Moreover, these matrices are not usually in standard form. Note that a generator polynomial of C is unique.

Corollary 9.5: generator polynomial of $C^\perp$

$C^\perp$ is also a cyclic code with generator polynomial $h_0^{-1}\overleftarrow{h}(x)$. (Scaling by h_0^{-1} is necessary because the generator polynomial must by definition be monic.)

Example: cyclic binary codes of length 3

Use Theorem 9.3 and Theorem 9.4 to find all the cyclic binary codes of length 3.

Solution. Generator polynomials are **monic factors of** $x^n - 1$ **in** $\mathbb{F}_q[x]$. The first step is to factorise $x^n - 1$ into **irreducible monic polynomials** in $\mathbb{F}_q[x]$. A polynomial is irreducible if it cannot be written as a product of two polynomials of positive degree.

Note that the polynomial $x^n - 1$ is **not** irreducible in $\mathbb{F}_q[x]$. Indeed, $x^n - 1 = (x-1)(x^{n-1} + \cdots + x + 1)$.

We work over the field $\mathbb{F}_2$ and observe:

$$x^3 - 1 = (x-1)(x^2 + x + 1).$$

The polynomial $x - 1 = x + 1$ is irreducible, because it is of degree 1.

Can we factorise the polynomial x^2+x+1 in $\mathbb{F}_2[x]$? If we could, we would have a factorisation $(x+a)(x+b)$. But then $ab = 1$ which means $a = b = 1$ in $\mathbb{F}_2$. Note that $(x+1)^2 = x^2 + 1$ in $\mathbb{F}_2[x]$. We have shown that $x^2 + x + 1$ is irreducible in $\mathbb{F}_2[x]$.

So the possible monic factors of $x^3 - 1$ in $\mathbb{F}_2[x]$ are:

$$1; \qquad 1 + x; \qquad 1 + x + x^2; \qquad 1 + x^3.$$

We now list every cyclic code in $\mathbb{F}_2^3$, giving its generator matrix G, minimum distance d and a well-known name of the code, and point out its dual code (which is also cyclic).

- $g(x) = 1$, $G = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ which corresponds to the **trivial binary code** of length 3: $C = \mathbb{F}_2^3$ with $d = 1$. The dual code of $\mathbb{F}_2^3$ is the null code (see below).

- $g(x) = 1 + x$, $G = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$. This is $\{000, 110, 011, 101\} = E_3$, the binary even weight code of length 3 which has $d = 2$. The dual of E_3 is $Rep(3, \mathbb{F}_2)$ (see below).

- $g(x) = 1 + x + x^2$, $G = \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$. This is $\{000, 111\} = Rep(3, \mathbb{F}_2)$, the binary repetition code of length 3 with $d = 3$. This code is $(E_3)^\perp$.

- $g(x) = 1 + x^3$. Theorem 9.4 returns matrix G with $k = 3 - 3 = 0$ rows, $G = [\quad]$. And indeed, by definition $1 + x^3$ is the generator polynomial of the null code $\{000\}$, which has empty generator matrix. It is a useless code but formally it is a linear and cyclic code, so we have to allow it for reasons of consistency. The minimum distance of the zero code is undefined. This code is $(\mathbb{F}_2^3)^\perp$.

Week 9

Exercises (answers at end)

Version 2023-11-10. To accessible online version of these exercises

Exercise 9.1. Find all cyclic codes of weight 1 in $\mathbb{F}_q^n$.

Exercise 9.2. A *burst* of length $\leq l$ is defined as a vector in $\mathbb{F}_q^n$ with chosen l consecutive symbols such that all non-zeros occur only within the chosen l symbols.

(a) Explain why a burst of length $\leq l$ has weight at most l, but not every vector of weight l or less is a burst of length $\leq l$.

(b) Let $C \subseteq \mathbb{F}_q^n$ be a cyclic code with generator polynomial of degree r. Show that C detects all burst errors of length $\leq r$. (*That is, a burst of length $\leq r$ is not a codevector.*)
Hint: cyclically shift $\underline{b}$ to positions $0, 1, \ldots, r-1$ so that the polynomial $b(x)$ is of degree $\leq r-1$. Show that such a polynomial cannot be in the code.

(*Informally: this means that burst error detection by cyclic codes is better than "generic" error detection. Cyclic codes are used on memory cards and in Ethernet networks where the errors that occur are likely to be burst errors — scratches, electrical noise etc.*)

Exercise 9.3. Data read from an SD card is encoded by CRC-16-CCITT: a binary cyclic code C with generator polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$. The least n for which $g(x)$ divides the polynomial $x^n - 1$ in $\mathbb{F}_2[x]$ is $n = 32767$; accordingly, C is of length 32767.

(a) Calculate the rate of C.

(b) Show that C detects all burst errors of length up to 16.

(c) Show that all codevectors of C have even weight. Hint: start with the codevector $\underline{g}$.

(d) Explain why $d(C) \leq 4$. Deduce from (c) that $d(C)$ is even. Prove that $d(C) = 4$.

Week 9

Exercises — solutions

Version 2023-11-10. To accessible online version of these exercises

Exercise 9.1. Find all cyclic codes of weight 1 in $\mathbb{F}_q^n$.

Answer to E9.1. If C is of weight 1, C contains a vector $(0, \ldots, 0, \lambda, 0, \ldots, 0)$ where λ is the only non-zero symbol; hence by linearity also $(0, \ldots, 0, 1, 0, \ldots, 0)$. Cyclic shifts of this vector span the space $\mathbb{F}_q^n$, so C must contain all vectors of length n. Hence C is trivial. Trivial codes are the only cyclic codes of weight 1.

Exercise 9.2. A *burst* of length $\leq l$ is defined as a vector in $\mathbb{F}_q^n$ with chosen l consecutive symbols such that all non-zeros occur only within the chosen l symbols.

(a) Explain why a burst of length $\leq l$ has weight at most l, but not every vector of weight l or less is a burst of length $\leq l$.

(b) Let $C \subseteq \mathbb{F}_q^n$ be a cyclic code with generator polynomial of degree r. Show that C detects all burst errors of length $\leq r$. (*That is, a burst of length $\leq r$ is not a codevector.*)
Hint: cyclically shift $\underline{b}$ to positions $0, 1, \ldots, r-1$ so that the polynomial $b(x)$ is of degree $\leq r-1$. Show that such a polynomial cannot be in the code.

(*Informally: this means that burst error detection by cyclic codes is better than "generic" error detection. Cyclic codes are used on memory cards and in Ethernet networks where the errors that occur are likely to be burst errors — scratches, electrical noise etc.*)

Answer to E9.2.

(a) Let $l \geq 2$. The vector $1, \underbrace{0, \ldots, 0}_{l-1 \text{ zeros}}, 1, 0, \ldots, 0$ is of weight $2 \leq l$ but is not a burst of length $\leq l$.

(b) Let $\underline{b} \neq 0$ be a burst of length $\leq r$. Assume for contradiction that $\underline{b}$ is a codeword of C. Since C is a cyclic code, all vectors obtained from $\underline{b}$ by cyclic shifts are in C. In particular, the following vector can be obtained from $\underline{b}$ by cyclic shifts:

$$\underline{b}' = \underbrace{b_0\, b_1\, \ldots\, b_{r-1}}_{r \text{ symbols}}\, 0\, 0\, \ldots\, 0,$$

where the last $n - r$ symbols are zero.

The codevector $\underline{b}'$ corresponds to the code polynomial $b_0 + b_1 x + \ldots + b_{r-1} x^{r-1}$ which must then be divisible by $g(x)$. But a non-zero polynomial of degree $\leq r - 1$ cannot be divisible by a polynomial of degree r, a contradiction.

Exercise 9.3. Data read from an SD card is encoded by CRC-16-CCITT: a binary cyclic code C with generator polynomial $g(x) = x^{16} + x^{12} + x^5 + 1$. The least n for which $g(x)$ divides the polynomial $x^n - 1$ in $\mathbb{F}_2[x]$ is $n = 32767$; accordingly, C is of length 32767.

(a) Calculate the rate of C.

(b) Show that C detects all burst errors of length up to 16.

(c) Show that all codevectors of C have even weight. Hint: start with the codevector $\underline{g}$.

(d) Explain why $d(C) \leq 4$. Deduce from (c) that $d(C)$ is even. Prove that $d(C) = 4$.

Answer to E9.3. (a) The dimension of C is $k = n - \deg g = 32751$ so the rate is $R = k/n = 32751/32767 \approx 0.9995$.

(b) Follows from part (b) of the previous exercise.

(c) Looking at the coefficients of the polynomial $g(x)$, we conclude that the codevector $\underline{g}$ has four 1s, in positions 0, 5, 12 and 16. Thus, $w(\underline{g}) = 4$.

A generator matrix of C has cyclic shifts of $\underline{g}$ as rows; so each row is of weight 4. It follows that all rows of G lie in the binary even weight code E_n. Since E_n is a linear code, all linear combinations of rows of G — that is, all codevectors of C — also lie in E_n and so have even weight.

(d) Let $\underline{c}$ be a codevector of C of minimum positive weight; that is, $w(C) = w(\underline{c})$. By (c), $w(\underline{c})$ is even. By minimality, $w(\underline{c}) \leq w(\underline{g}) = 4$. It follows that $w(\underline{c})$ is either 2 or 4.

Assume for contradiction that $w(\underline{f}) = 2$. Up to a cyclic shift, the vector $\underline{c}$ (which has two non-zero bits) corresponds to code polynomial of the form $1 + x^d$ for some $d < 32767$. But all code polynomials are divisible by $g(x)$, and we are given that $g(x)$ divides $1 + x^n$ for $n = 32767$ but not for any smaller n. This is a contradiction which proves that $w(\underline{c}) = 4$.

General remark: *Although the generator matrix and the parity check matrix of this code are large, the encoding and error detection algorithms are based on polynomial division with remainder. This has efficient hardware and software implementations.*

Week 10

Golay codes. Classification of perfect codes

Version 2023-11-29. To accessible online version of this chapter

Synopsis. *One can explore cyclic codes of a given length over a given finite field in an attempt to find codes with interesting/useful properties. In fact, all types of codes we have considered so far will arise as cyclic codes. In this chapter, we define two new linear equivalence classes of codes called Golay codes. In our approach, these arise as cyclic codes, however, historically they were found in a different way. We give without proof a complete classification of perfect codes over alphabets of prime power size up to parameter equivalence, conjectured by Golay and proved by Tietäväinen and van Lint.*

Recall that:

- the only way to specify a general non-linear code in $\mathbb{F}_q^n$ is to list all the codewords, which consist of a total of $q^k \times n$ symbols;
- a linear code can be specified by a generator matrix, which has $k \times n$ entries;
- a cyclic code can be specified in an even more compact way — by giving its generator polynomial, which corresponds to a single codeword! We only need to specify $n-k$ coefficients of the generator polynomial (its degree is $n-k$ and its leading coefficient is 1).

Approach to searching for interesting/perfect/etc codes:

Look for divisors of $x^n - 1$ and hope that the cyclic codes they generate have a large minimum distance. **For example,** among the cyclic codes in $\mathbb{F}_2^7$, there are two perfect, Hamming codes (*Exercise*).

We will now describe two codes found by Marcel Golay in 1949. They are known as the *binary Golay code* G_{23} and the *ternary Golay code* G_{11}, respectively.

The binary Golay code G_{23}

In $\mathbb{F}_2[x]$, $x^{23}-1=(x+1)g(x)\overleftarrow{g}(x)$, where $g(x)=x^{11}+x^{10}+x^6+x^5+x^4+x^2+1$ and $\overleftarrow{g}(x)=x^{11}+x^9+x^7+x^6+x^5+x+1$.

Exercise: check this! You may use a computer algebra system but it is always instructive to multiply these out by hand.

Definition: binary Golay code G_{23}

Define a **binary Golay code** to be the cyclic code in $\mathbb{F}_2^{23}$ generated by $g(x)$, or any code linearly equivalent to it. (Any) binary Golay code is denoted G_{23}.

Remark: The cyclic code generated by $\overleftarrow{g}(x)$ is seen to be linearly equivalent to the cyclic code generated by $g(x)$; the linear equivalence is by writing all the codevectors backwards.

The above definition does not reflect how the code was originally found (see below) but suggests a practical way to construct a G_{23} code if need be: factorise $x^{23}-1$ over $\mathbb{F}_2$ into irreducible factors (e.g., using a computer algebra system) and take one such factor of degree greater than 1 to be the generator polynomial of a cyclic code.

Theorem 10.1: parameters of G_{23}

G_{23} is a perfect $[23,12,7]_2$-code.

Proof of Theorem 10.1 — part 1. The code is binary ($q=2$) of length $n=23$ by construction. The dimension is $k=23-\deg g=12$.

It is easy to see that the weight of G_{23} is **at most** 7: indeed, the vector $\underline{g}\in G_{23}$ is 10101110001100000000000, of weight 7, and so $w(G_{23})\le 7$.

It is more difficult to show that the weight of G_{23} is exactly 7. We will present a theoretical proof of this result using the extended code G_{24}, and will also show how to obtain the same result by a computer calculation.

We now prove that a $[23,12,7]_2$ -code is perfect. The Hamming bound for a binary code in in logarithmic form is $k\le n-\log_2\left(\binom{n}{0}+\cdots+\binom{n}{t}\right)$. Here $t=[(7-1)/2]=3$ so the argument of $\log_2$ is $1+\binom{23}{1}+\binom{23}{2}+\binom{23}{3}=1+23+23\times\frac{22}{2}+23\times\frac{22}{2}\times\frac{21}{3}=1+23(1+11+77)=2048$. One has $12=23-\log_2 2048$ hence the Hamming bound is attained.

The proof that $w(G_{23})=7$ will be given after a series of lemmas **(proof to be continued).**

Binary vectors: extending, overlaps, weights and orthogonality

To proceed, we need a mini-toolbox containing tools for working with binary vectors.

A binary vector $\underline{v} = (v_1, v_2, \ldots, v_n)$ is **extended** to obtain the vector $\widehat{\underline{v}} = (v_1, \ldots, v_n, v_{n+1})$ where $v_{n+1} = v_1 + \cdots + v_n$ in $\mathbb{F}_2$. That is, a vector is extended by appending one bit so that the resulting vector has even weight. Explicitly, we may write

$$\widehat{\underline{v}} = \begin{cases} (\underline{v}, 0), & \text{if } w(\underline{v}) \text{ is even,} \\ (\underline{v}, 1), & \text{if } w(\underline{v}) \text{ is odd.} \end{cases}$$

By extending each vector in a given binary code, we obtain the *extended code*:

Definition: extended code

If C is a binary linear code of length n, we define the **extended code** $\widehat{C}$ of length $n+1$ as $\{\widehat{\underline{c}} : \underline{c} \in C\}$.

The following notion is useful:

Definition: overlap

If $\underline{u}, \underline{v} \in \mathbb{F}_2^n$, the **overlap** of $\underline{u}$ and $\underline{v}$ is the number of positions i such that $u_i = v_i = 1$.

It is easy to see that

$$w(\underline{u} + \underline{v}) = w(\underline{u}) + w(\underline{v}) - 2 \times \text{overlap}(\underline{u}, \underline{v}) \tag{10.1}$$

and

$$\underline{u} \cdot \underline{v} = 0 \quad \iff \quad \text{overlap}(\underline{u}, \underline{v}) \text{ is even.} \tag{10.2}$$

It follows that

$$w(\underline{u}), w(\underline{v}) \text{ are multiples of } 4,\ \underline{u} \cdot \underline{v} = 0 \quad \implies \quad w(\underline{u} + \underline{v}) \text{ is a multiple of } 4. \tag{10.3}$$

Indeed, by (10.1), $w(\underline{u}+\underline{v})$ is (multiple of 4) $+$ (multiple of 4) $- 2\times$ overlap($\underline{u}$, $\underline{v}$), and by (10.2), $2\times$ overlap($\underline{u}$, $\underline{v}$) is a multiple of 4 so the result is a multiple of 4.

The extended binary Golay code G_{24}

Definition: the extended binary Golay code G_{24}

The extended code $\widehat{G}_{23}$ is called the **extended binary Golay code** and is denoted G_{24}.

The code G_{24} is not cyclic, but we can modify the cyclic code methods used for G_{23} to answer questions about G_{24}. For example:

Example: generator matrix for G_{24}

Write down a generator matrix for G_{24}.

Solution. Theorem 9.4 gives a generator matrix for G_{23} as follows: the top row is the vector $\underline{g} = 10101110001100000000000$, and the rest of the rows are its cyclic shifts $\underline{xg}, \ldots, \underline{x^{11}g}$. Extending each of these rows (of weight 7 which is odd) by appending 1 gives twelve codevectors of G_{24}, forming the matrix

$$G = \begin{bmatrix}
10101110001100000000000\mathbf{1}\\
01010111000110000000000\mathbf{1}\\
00101011100011000000000\mathbf{1}\\
00010101110001100000000\mathbf{1}\\
00001010111000110000000\mathbf{1}\\
00000101011100011000000\mathbf{1}\\
00000010101110001100000\mathbf{1}\\
00000001010111000110000\mathbf{1}\\
00000000101011100011000\mathbf{1}\\
00000000010101110001100\mathbf{1}\\
00000000001010111000110\mathbf{1}\\
00000000000101011100011\mathbf{1}
\end{bmatrix}$$

The rows of G are linearly independent, because they give a linearly independent set if you delete the last bit). By definition of extended code, $\#G_{24} = \#G_{23} = 2^{12}$ and so $\dim G_{24} = 12$, same as the number of rows of G. Hence G is a generator matrix for G_{24}.

The next two propositions establish two main properties of G_{24}.

Proposition 10.2: G_{24} is self-dual

G_{24} is a self-dual code, that is, $G_{24} = G_{24}^{\perp}$.

Proof. It is enough to check that the above generator matrix G for G_{24} satisfies $GG^T = 0$ — that is, its rows $\underline{r}_0, \ldots, \underline{r}_{11}$ are orthogonal to each other — and $n = 2k$. The latter is clear as $24 = 2 \times 12$. The former can be done in two ways.

Way 1 (manual): recall (10.2). Manually check that the overlap of $\underline{r}_i$ and $\underline{r}_j$ is even for all i, j. It is enough to check the overlap of the top row with the other rows — the rest follows by cyclic shifts of the first 23 bits.

Way 2 (working with polynomials): Write the rows of G as $\underline{r}_i = \widehat{\underline{x^i g}} = (\underline{x^i g}, 1)$ for $i = 0, 1, \ldots, 11$. We calculate the inner product, $(\underline{x^i g}, 1) \cdot (\underline{x^j g}, 1) = \underline{x^i g} \cdot \underline{x^j g} + 1$, of two rows of G. Recall from the proof of Theorem 9.4 that the inner product of vectors $\underline{a}$ and $\overleftarrow{\underline{b}}$

is the coefficient of x^{n-1} in the polynomial $a(x)b(x)$. The vector $\underline{x^j g}$ written backwards is seen to be $\underline{x^{11-j}\overleftarrow{g}(x)}$, so

$$\underline{x^i g} \cdot \underline{x^j g} + 1 = \big(\text{coef. of } x^{22} \text{ in } x^{i+11-j} g(x)\overleftarrow{g}(x)\big) + 1.$$

Note that

$$g(x)\overleftarrow{g}(x) = \frac{x^{23}-1}{x-1} = x^{22} + x^{21} + \cdots + x + 1$$

is a polynomial where the coefficients of $x^0, \ldots, x^{22}$ are all 1 and so x^{22} appears in the polynomial $x^{i+11-j} g(x)\overleftarrow{g}(x)$ with coefficient 1. Thus, $\underline{x^i g} \cdot \underline{x^j g} + 1 = 1 + 1 = 0$. $\square$

Self-duality of G_{24} allows us to deduce other further properties of this code.

Proposition 10.3: weights in G_{24}

The weight of every codevector of G_{24} is a multiple of 4.

Proof. Each row $\underline{r}_i$ of the generator matrix G constructed in the proof of Proposition 10.2 has weight 8 which is a multiple of 4. By Proposition 10.2, rows of G are mutually orthogonal, so by (10.3), a sum $\underline{r}_i + \underline{r}_j$ of two rows of G also has weight divisible by 4.

We can now apply (10.3) to a sum of $\underline{r}_i + \underline{r}_j$ and $\underline{r}_k$ (both are codevectors of G_{24} so their inner product is zero by Proposition 10.2) to show that a sum of three rows of G has weight divisible by 4. Continuing in the same way, we show that a sum of any number of rows of G, i.e., any codevector of G_{24}, has weight divisible by 4. $\square$

Remark: rest of proof of Theorem 10.1

We are now ready to finish the proof of Theorem 10.1 about the parameters of G_{23}.

Proof of Theorem 10.1 — part 2 (final). We are left to prove that the binary Golay code G_{23} does not contain non-zero codevectors of weight less than 7.

We will take G_{23} to be cyclic with generator polynomial $g(x)$, and will interchangeably use vectors and polynomials. Assume $\underline{v} \in G_{23}$. If a vector $\underline{v}'$ obtained from $\underline{v}$ by applying the cyclic shift m times, then $v' \in G_{23}$; note that a term x^i in the polynomial $v(x)$ is shifted to x^{i+m} in $v'(x)$ if $i + m < n$, more generally to $(i+m) \bmod n$, where $n = 23$.

$w(\underline{v})$ **cannot be 1, 2, 5 or 6.** If $\underline{v} \in G_{23}$ has weight 1, 2, 5 or 6, then the extended vector $\widehat{\underline{v}} \in G_{24}$ has weight 2 or 6, not divisible by 4, contradicting Proposition 10.3.

$w(\underline{v})$ **cannot be 3.** Assume $w(\underline{v}) = 3$ so that $v(x) = x^i + x^j + x^k$. Out of the 22 possible cyclic shifts of $\underline{v}$, at most six can have non-zero overlap with $\underline{v}$: these shift x^a to x^b for some $a, b \in \{i, j, k\}$. Hence there exists a shift $\underline{v}'$ of $\underline{v}$ which has zero overlap with $\underline{v}$. Then $\underline{v} + \underline{v}' \in G_{23}$ has weight 6, contradicting the previous case.

$w(\underline{v})$ **cannot be 4.** Suppose it can, and shift $\underline{v}$ so that $v(x) = 1 + x^a + x^b + x^c$ with $0 < a < b < c$. Pick a code polynomial of weight 4 of least possible degree c.

Shifting $v(x)$ to the left a times gives $v'(x) = 1 + x^{b-a} + x^{c-a} + x^{n-a}$. Note that $(\underline{v}, 0)$ and $(\underline{v}', 0)$ lie in $G_{24} = G_{24}^{\perp}$ and so have inner product 0, hence $\underline{v} \cdot \underline{v}' = 0$ and by (10.2) the overlap of $\underline{v}$ and $\underline{v}'$ must be even. The overlap is not 4 because $v'(x) \neq v(x)$: otherwise one would have $b = 2a$, $c = 3a$ and $n = 4a$, impossible as $n = 23$. The overlap is not 0 as $v(x)$ and $v'(x)$ have term 1 in common. Hence the overlap of $\underline{v}$ and $\underline{v}'$ is 2.

Observe that $n - a = c$ is impossible, as it would give the code polynomial $v(x) - v'(x)$ of degree less than c and weight 2 (impossible by earlier cases) or 4 (contradicts minimality of c), so $n - a > c$. Neither x^{n-a} nor x^c contribute to the overlap of $\underline{v}$ and $\underline{v}'$, which leaves three cases of how overlap 2 could be achieved.

Case $c - a = b$**.** Then $v(x) = 1 + x^a + x^b + x^{a+b}$ which factorises as $(1 + x^a)(1 + x^b)$. The code polynomial $v(x)$ is divisible by the generator polynomial $g(x)$ which is irreducible, so $1 + x^a$ or $1 + x^b$ must be divisible by $g(x)$. But this means a codevector of weight 2, a contradiction.

Case $c - a = a$**.** Writing $b = a + d$, we have $v(x) = 1 + x^a + x^{a+d} + x^{2a}$. Shift d times to obtain $v''(x) = x^d + x^{a+d} + x^{a+2d} + x^{2a+d}$ (since $2a < n - a$, we have $2a + d < n$). The polynomials $v(x)$ and $v''(x)$ have the term x^{a+d} in common, and the only possibility for the overlap of $\underline{v}$ and $\underline{v}''$ to be 2 is $a + 2d = 2a$, that is, $a = 2d$. Then $v(x) = 1 + x^{2d} + x^{3d} + x^{4d}$ which factorises as $(1 + x^d)(1 + x^d + x^{3d})$. As above, either $1 + x^d$ or $1 + x^d + x^{3d}$ must be divisible by $g(x)$, so there is a codevector of weight 2 or 3, a contradiction.

Case $b - a = a$**.** We have $v(x) = 1 + x^a + x^{2a} + x^c$, and shift $2a$ times gives $v''(x) = x^{2a} + x^{3a} + x^{4a} + x^{(c+2a) \bmod n}$. The overlap of $\underline{v}$ and $\underline{v}''$ must be 2, and the two polynomials have the term x^{2a} in common, so there must be another common term. This is only possible in two subcases.

Subcase $c = 4a$**.** We have $v(x) = 1 + x^a + x^{2a} + x^{4a}$ which factorises as $(1 + x^a)(1 + x^{2a} + x^{3a})$. As above, this means a codevector of weight 2 or 3, contradicting earlier results.

Subcase $(c+2a) \bmod n = 0$**.** Here we have the code polynomial $v''(x) = 1 + x^{2a} + x^{3a} + x^{4a}$, which factorises in the same way as in the case $c - a = a$ above, so that we arrive at the same contradiction.

Conclusion. We showed that G_{23} has no codevectors of weight $1, 2, 3, 4, 5, 6$ and so $w(G_{23}) \geq 7$ as claimed. This completes the proof of Theorem 10.1. $\square$

The above theoretical proof that $w(G_{23}) = 7$ gives the taste of how Coding Theory was done in the last century. Today, the weight of G_{23} can be easily found using a computer —

consider for example the following code written for the computer algebra system **SageMath**:

```
sage: R.<x>=GF(2)[]
sage: factor(x^23 - 1)
(x+1)*(x^11+x^9+x^7+x^6+x^5+x+1)*(x^11+x^10+x^6+x^5+x^4+x^2+1)
sage: g = factor(x^23 - 1)[1][0]
sage: messagepolynomials = R.monics( max_degree=23-g.degree()-1 )
sage: codepolynomials = [ u*g for u in messagepolynomials ]
sage: min([ len(c.coefficients()) for c in codepolynomials ])
7
```

Is the above code a proof? Many mathematicians would accept it as the source code can be checked, and the calculation reproduced.

Remark: trivia

The code G_{24} was used by Voyager 1 & 2 spacecraft to transmit information back to Earth (NASA, Jupiter and Saturn, 1979–81).

The ternary Golay code G_{11}

In $\mathbb{F}_3[x]$, $x^{11}-1=(x-1)g(x)g_1(x)$ where $g(x)=x^5+x^4+2x^3+x^2+2$ and $g_1(x)=-\overleftarrow{g}(x)=x^5+2x^3+x^2+2x+2$.

Definition: the ternary Golay code G_{11}

A **ternary Golay code** is the the cyclic code in $\mathbb{F}_3^{11}$ generated by $g(x)$, or any code linearly equivalent to it. (*Notation*: G_{11}.)

Theorem 10.4: paremeters of G_{11}

G_{11} is a perfect $[11,6,5]_3$ code.

The crucial, and difficult, step in a theoretical proof of Theorem 10.4 is showing that G_{11} does not contain non-zero codevectors of weight less than 5. We omit the proof.

An alternative approach is a computer-based calculation:

Exercise. Prove Theorem 10.4, modifying the computer code provided after the proof of Theorem 10.1 to calculate the weight of G_{11}.

Historical notes

Golay found his two perfect codes in 1949, before cyclic codes were discovered. He wrote check matrices for G_{23} and G_{11}. Crucially, Golay observed that $\binom{23}{0}+\binom{23}{1}+\binom{23}{2}+\binom{23}{3}$

is a power of two. From the proof of perfectness above one can see that the condition $\binom{n}{0}+\cdots+\binom{n}{t}=2^r$ is necessary for the existence of a perfect t-error-correcting binary code of length n. This condition is not sufficient: e.g., in his 1949 paper Golay also observes that $\binom{90}{0}+\binom{90}{1}+\binom{90}{2}=2^{12}$ but this does not lead to any perfect binary code of length 90.

Amazingly, Golay's 1949 paper where he constructs all the Hamming codes and the two Golay codes, is barely half a page long.

Now we can state the classification result about perfect codes.

Definition: parameter equivalence

We say that two codes are **parameter equivalent**, if they both are $[n,k,d]_q$-codes for some n,k,d and q.

The following theorem was proved by Tietäväinen and van Lint in 1973, more than twenty years since Golay gave a conjectural list of perfect codes in alphabets of prime power size. We will not give its proof here, but you should learn the statement of the theorem.

Theorem 10.5: classification of perfect codes where q is a prime power

Let q be a power of a prime number. A perfect $[n,k,d]_q$-code is parameter equivalent to one of the following:

- a trivial code: n arbitrary, $k=n$, $d=1$, q any prime power;
- a binary repetition code of odd length: n odd, $k=1$, $d=n$, $q=2$;
- a Hamming code $\mathrm{Ham}(r,q)$: $n=\dfrac{q^r-1}{q-1}$, $k=n-r$, $d=3$, q any prime power;
- the Golay code G_{23}, which is a $[23,12,7]_2$-code;
- the Golay code G_{11} which is an $[11,6,5]_3$-code.

Remark: perfect codes over general alphabets

Classification of perfect codes over alphabets of size not equal to a prime power is, in general, an open problem.

Week 10

Exercises (answers at end)

Version 2023-11-29. To accessible online version of these exercises

Exercise 10.1 (the extended binary Golay code)**.** (a) Determine the parameters $[n, k, d]_q$ of G_{24}. State how many bit errors per codevector is the code guaranteed to *detect*. Same for *correct*. Find the rate of G_{24}.

(b) A codevector of G_{24} is transmitted, and thirteen bit errors occur. Will an error be detected?

Exercise 10.2 (*This exercise is discussed in the review sessions*)**.** Find all possible binary cyclic codes of length 7. For each such code, find its minimum distance, determine whether the code is perfect. Determine which codes that you obtain are linearly equivalent.

Exercise 10.3. (i) Show that a perfect ternary code of length 11 and minimum distance 5 must contain 729 codewords.

(ii) A football match can end in a Win (2), Draw (1) or Loss (0) for your club. You buy a *football pool* ticket which contains 11 boxes. You fill in the boxes trying to predict the result of each of the 11 matches your club will play in a forthcoming tournament. If, at the end of the tournament, it turns out that your ticket contained 9 or more correct guesses (out of 11), you win a prize.

(a) Assuming that the outcomes of the 11 matches are completely independent and random, show that one ticket wins a prize with a probability $\frac{1}{729}$. [*Of course, this does not mean that just by completing 729 tickets you are guaranteed a prize!*]

(b) Explain how one can use a code from (i) to buy and complete 729 football pool tickets and to *guarantee* that one of them wins a prize.

Week 10

Exercises — solutions

Version 2023-11-29. To accessible online version of these exercises

Exercise 10.1 (the extended binary Golay code)**.** (a) Determine the parameters $[n, k, d]_q$ of G_{24}. State how many bit errors per codevector is the code guaranteed to *detect*. Same for *correct*. Find the rate of G_{24}.

(b) A codevector of G_{24} is transmitted, and thirteen bit errors occur. Will an error be detected?

Answer to E10.1. Extending a binary code means appending a parity check bit to every codevector, so that the resulting vector is of even weight. Appending one bit increases the length by 1, so the length n of G_{24} is 24.

Appending a bit to every codevector does not change the number of codevectors. The G_{23} is a $[23, 12, 7]_2$ code, so $\#G_{24} = \#G_{23} = 2^{12}$ and $k = 12$.

The minimum weight vector in $G_{23} \setminus \{0\}$ has weight 7, so after extending, it becomes a vector of weight 8. Extending a vector cannot decrease its weight, so $d = w(G_{24}) = 8$. A $[24, 12, 8]_2$ code which is guaranteed to detect up to $8 - 1 = 7$ errors per codevector and is guaranteed to correct $[(8 - 1)/2] = 3$ errors per codevector.

The rate of G_{24} is $k/n = 12/24 = 1/2$.

(b) When 13 bit errors occur in a vector of even weight, the received vector has odd weight. Since all codevectors of G_{24} have even weight (by construction), this will result in a detected error — despite 13 being greater than 7.

Exercise 10.2 (*This exercise is discussed in the review sessions*)**.** Find all possible binary cyclic codes of length 7. For each such code, find its minimum distance, determine whether the code is perfect. Determine which codes that you obtain are linearly equivalent.

Answer to E10.2. First of all, one needs to write the polynomial $x^7 - 1$ as a product of irreducible factors. One can always start with $x - 1$, because $x - 1$ is always a factor of $x^n - 1$ for any n and for every field $\mathbb{F}_q$. We have $x^7 - 1 = (x-1)(x^6 + x^5 + x^4 + x^3 + x^2 + x + 1)$.

We now need to factorise $x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$ over $\mathbb{F}_2$. Unfortunately there is no easy way. One can use brute force (*this will not be expected in the exam*): check whether any of the polynomials of degree 1 are factors of $x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$, then polynomials of degree 2, then polynomials of degree 3. In this case we obtain the factorisation $x^6 + x^5 + x^4 + x^3 + x^2 + x + 1 = (x^3 + x + 1)(x^3 + x^2 + 1)$. We conclude that

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

as a product of irreducible polynomials over $\mathbb{F}_2$. (Note that $x - 1$ is the same as $x + 1$ over $\mathbb{F}_2$.)

There are thus 8 cyclic binary codes of length 7: they correspond to generator polynomials which are product of a subset of the three irreducible factors of $x^7 - 1$. To list all of them, we denote $g_1 = x + 1$, $g_2 = x^3 + x + 1$, $g_3 = x^3 + x^2 + 1$:

g (generator polynomial)	$\deg g$	$\dim gR_7$
1	0	7
g_1	1	6
g_2, g_3	3	4
g_1g_2, g_1g_3	4	3
g_2g_3	6	1
$g_1g_2g_3$	7	0

The code of dimension 0 is $\{\underline{0}\}$, a cyclic code with generator polynomial $x^7 - 1$.

Dimension 1: the generator polynomial of this code is $(x^3 + x + 1)(x^3 + x^2 + 1) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$, see the table given above. Therefore, its generator matrix is

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

This is the generator matrix of the *binary repetition code of length* 7. The minimum distance of this code is 7.

Dimension 6: according to the table, the generator polynomial is $g = x + 1$. Therefore, the check polynomial is $h = \frac{x^7 - 1}{g} = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$. Then the check matrix is $H = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$. The check matrix generates the dual code, which therefore is the repetition code. The code of dimension 6 is dual to the repetition code, hence is the *binary even weight code of length* 7. The minimum distance is 2.

Dimension 7: the only code which has dimension equal to length is the trivial code. So the answer is the *trivial binary code of length* 7. It has generator polynomial 1 (of degree 0).

Dimension 3: we get two codes which are seen to be *simplex codes* $\Sigma(3,2) = \mathrm{Ham}(3,2)^\perp$. From the above factorisation, there are two generator polynomials of degree 4:

$g_1(x) = (x+1)(x^3+x+1) = x^4+x^3+x^2+1$ and $g_2(x) = (x+1)(x^3+x^2+1) = x^4+x^2+x+1$,

giving rise to the generator matrices

$$G_1 = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \quad \text{and} \quad G_2 = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 & 1 \end{bmatrix}$$

which generate the codes C_1 and C_2. Observing that G_2 is obtained from G_1 by permuting columns (e.g., permutation $2 \to 6 \to 4 \to 2$, $3 \leftrightarrow 5$ *or* notice that both matrices are made up of *all* possible non-zero columns of size 3 — they are parity check matrices for Hamming code $\mathrm{Ham}(3,2)$), we conclude that both codes are $\Sigma(3,2)$ and are linearly equivalent. Recall that their weight can be found by writing down all the 7 non-zero codevectors; all of them have weight 4. They are $[7,3,4]_2$-codes and are not perfect (the minimum distance is even).

Dimension 4: there are two codes, one generated by x^3+x+1, the other by x^3+x^2+1. Consider the code D with generator polynomial $g(x) = x^3+x+1$. The parity check polynomial of D is $h(x) = (x+1)(x^3+x^2+1) = x^4+x^2+x+1$ so its parity check matrix given by Theorem 9.4 is

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

This is the same as matrix G_1 above (with the order of rows reversed — but this does not affect the code generated by the matrix), hence $D^\perp$ is a $\Sigma(3,2)$ code C_1. Therefore, D is a $\mathrm{Ham}(3,2)$ code, which is a perfect $[7,4,3]_2$-code.

A completely similar argument shows that the code D' with generator polynomial x^3+x^2+1 is dual to C_2, hence is another $\mathrm{Ham}(3,2)$ code and is linearly equivalent to D.

There are thus 8 binary cyclic codes of length 7. None of them has dimension 2 or 5.

Exercise 10.3. (i) Show that a perfect ternary code of length 11 and minimum distance 5 must contain 729 codewords.

(ii) A football match can end in a Win (2), Draw (1) or Loss (0) for your club. You buy a *football pool* ticket which contains 11 boxes. You fill in the boxes trying to predict the result of each of the 11 matches your club will play in a forthcoming tournament. If, at the end of the tournament, it turns out that your ticket contained 9 or more correct guesses (out of 11), you win a prize.

(a) Assuming that the outcomes of the 11 matches are completely independent and random, show that one ticket wins a prize with a probability $\frac{1}{729}$. [*Of course, this does not mean that just by completing 729 tickets you are guaranteed a prize!*]

(b) Explain how one can use a code from (i) to buy and complete 729 football pool tickets and to *guarantee* that one of them wins a prize.

Answer to E10.3. (i) Here is a calculation of the Hamming bound for a ternary code of length 11 and minimum distance 5: $t = [\frac{5-1}{2}] = 2$, $\#S_2(\underline{0}) = \binom{11}{0} + \binom{11}{1}(3-1) + \binom{11}{2}(3-1)^2 = 1 + 11 \times 2 + 55 \times 2^2 = 243 = 3^5$, so that the Hamming bound (hence the cardinality of a perfect code) is $3^{11} \,/\, \#S_2(\underline{0}) = 3^{11}/3^5 = 3^6 = 729$.

(ii) (a) Let $\underline{X}$ denote the vector of match outcomes. Let $\underline{Y}$ denote the vector of values written on the ticket. The probablility that $\underline{Y}$ wins a prize is the probability that $d(\underline{X}, \underline{Y}) \le 2$, or, the same, that $\underline{X}$ belongs to the sphere $S_2(\underline{Y})$. Given the assumption that $\underline{X}$ is uniformly distributed in $\mathbb{F}_3^{11}$, this probability is calculated as $\frac{\#S_2(\underline{Y})}{\#\mathbb{F}_3^{11}}$. Note that $\#S_2(\underline{Y}) = \#S_2(\underline{0}) = 243$, so that the answer is $243/3^{11} = 1/729$.

(b) In fact, this is how the ternary Golay code G_{11}, which is a perfect $[11, 6, 5]_3$ code, was discovered by Finnish football pool enthusiast Juhani Virtakallio in 1947. Read about this in:

A. Barg, *At the Dawn of the Theory of Codes*, The Mathematical Intelligencer 15, no. 1, 1993, pp. 20–26; `http://www.ece.umd.edu/~abarg/reprints/dawn.pdf`

Virtakallio published the code — all the 729 codewords — in three (!) issues of a football pool magazine. When Marcel Golay rediscovered the code in 1949, he realised that G_{11} is a linear code, so it is enough to give only a check matrix. Following the introduction of cyclic codes in 1957 by Eugene Prange, we can define this code by its generator polynomial $x^5 + x^4 + 2x^3 + x^2 + 2$.

Briefly, one should write the 729 codewords of this perfect code C in the 729 tickets. Recall from the proof of the Hamming bound that, since C is perfect, the space $\mathbb{F}_3^{11}$ is covered by spheres of radius $t = 2$ centred at codevectors from C. Hence every vector in $\mathbb{F}_3^{11}$ is at distance ≤ 2 from a codevector of C. Therefore, for every possible vector $\underline{X}$ of 11 match outcomes there will be one out of the 729 tickets (codewords) which will differ from $\underline{X}$ in at most two positions. That ticket will win the prize.

Week 11

Reed-Muller codes

Version 2023-12-03. To accessible online version of this chapter

Synopsis. *The minimum distance of a perfect code cannot exceed* 7 *unless the code is a repetition code. This is disappointingly low. In this final part of the course, we construct Reed-Muller codes, a family of codes with large minimum distance. Unfortunately, they are not perfect. The construction is based on Boolean functions, which arise in elementary logic as columns of truth tables and are used in cicruit design.*

Boolean functions

Fix $m \ge 1$. Denote by V^m the set of all binary words of length m. (It is the same as $\mathbb{F}_2^m$ but viewed without any vector space structure).

For example, V^3 is the set $\{000, 001, 010, 011, 100, 101, 110, 111\}$.

Definition: Boolean functions

A **Boolean function** is a (set-theoretical) function $f\colon V^m \to \mathbb{F}_2$.

Remark: the number of Boolean functions

The total number of all Boolean functions on V^m is $|\mathbb{F}_2|^{|V_m|} = 2^{2^m}$.

Remark: Boolean functions as rows of a truth table. One has certainly met Boolean functions when constructing truth tables for statements in basic logic. To give an illustration, let $m = 3$. Consider statements which involve variables x_1, x_2, x_3, each of which can take values 0 ($\textsc{false}$) or 1 ($\textsc{true}$).

We will represent a logical statement by a *row* (not column) in a truth table. (We use rows because it is common in Coding Theory to think of codevectors as of row vectors; and in

Reed-Muller codes, codevectors arise from functions.) In our example ($m = 3$), the table will have 8 columns:

x_1	0	1	0	1	0	1	0	1
x_2	0	0	1	1	0	0	1	1
x_3	0	0	0	0	1	1	1	1
$(x_1 \text{ and } x_2) \implies x_3$	1	1	1	0	1	1	1	1
$\mathbf{0}$	0	0	0	0	0	0	0	0
$\mathbf{1}$	1	1	1	1	1	1	1	1
v_2v_3	0	0	0	0	0	0	1	1

In this table, $(x_1 \text{ and } x_2) \implies x_3$ is a statement whose truth value depends on the values of x_1, x_2 and x_3. Therefore, it can be viewed as a Boolean function: its value at the binary word 000 is 1, at the word 100 the value is 1, and so on. The only binary word where this function takes the value 0 is the word 110: indeed, if x_1 and x_2 are TRUE, then x_1 and x_2 is TRUE, but x_3 is FALSE, and the value of the implication "TRUE $\implies$ FALSE" is FALSE.

(The other rows in the table will be explained below.)

The Boolean algebra

Because Boolean functions take values in $\mathbb{F}_2 = \{0, 1\}$ which is a field, Boolean functions can be added and multiplied pointwise: if $f, g \colon V^m \to \mathbb{F}_2$, one has the functions

$$f + g, fg \colon V^m \to \mathbb{F}_2; \quad (f + g)(x) = f(x) + g(x), \quad (fg)(x) = f(x)g(x), \quad \forall x \in V^m.$$

Also, there are constant functions $\mathbf{0}$ and $\mathbf{1}$. (They are shown in the 2nd, respectively 3rd, row of the truth table above.) The Boolean function $\mathbf{1}$ is often called *the tautological truth*.

Definition: Boolean algebra

The vector space of Boolean functions $f \colon V^m \to \mathbb{F}_2$, together with the operation of multiplication of functions, is the **Boolean algebra** on V^m.

The traditional logical operations can be written in terms of the Boolean algebra operations $+$ and $\times$. Clearly, multiplication is the same as AND:

$$fg = f \text{ and } g.$$

The addition obeys the rule $0 + 0 = 0$, $0 + 1 = 1 + 0 = 1$, $1 + 1 = 0$. The logical operation which corresponds to addition is called the *exclusive OR*:

$$f + g = f \text{ xor } g = ((f \text{ or } g) \text{ and } \text{not}(f \text{ and } g)).$$

How to write elements of the Boolean algebra as row vectors?

To write elements of the Boolean algebra on V^m as binary vectors, so that we can define the weight, the Hamming distance etc, we need to order all binary words of length m as $b_0, \ldots, b_{2^m-1}$.

The standard ordering is obtained by interpreting the word $x_1x_2\ldots x_m$ as a number written in base 2, i.e., the number $2^{m-1}x_1 + \ldots + 2x_{m-1} + x_m$. Thus, the binary words of length 3 appear in the following order: 000, 001, 010, 011, 100, 101, 110, 111. However, the exact choice of the order is not important, as we will see.

Definition: value vector of a Boolean function

Let $f\colon V^m \to \mathbb{F}_2$ be a Boolean function. The **value vector** of f is the binary vector $\underline{f} = (f(b_0), \ldots, f(b_{2^m-1}))$ of length 2^m, where $b_0, \ldots, b_{2^m-1}$ is the chosen ordering of V^m.

The next notion does not at all depend on the chosen ordering of words in V^m:

Definition: weight of a Boolean function

The **weight** of the Boolean function f is defined as the weight of the value vector $\underline{f}$. The weight does not depend on the ordering of the binary words, because

$$w(f) = \#\{b \in V^m : f(b) = 1\}.$$

The monomial basis of the Boolean algebra

We will now introduce two special kinds of elements of the Boolean algebra: coordinate functions and, more generally, monomial functions.

Definition: coordinate function

The Boolean function $v_i\colon V^m \to \mathbb{F}_2$ defined by $v_i(x_1, x_2, \ldots, x_m) = x_i$ is called the ith **coordinate function**.

Definition: monomial, polynomial, degree

To each subset $\{i_1, \ldots, i_r\} \subseteq \{1, \ldots, m\}$ there corresponds the **monomial function** (or **monomial**) $v_{i_1} \ldots v_{i_r}$, of **degree** r.
Also, $\mathbf{1}$ is the monomial function corresponding to the set $\varnothing$, of degree 0.
A linear combination of monomials is a **polynomial**. The degree of a polynomial f is the highest degree of a monomial which appears in f.

Remark: properties of monomials.

- Observation: because the values of any Boolean function are 0 and 1, one has $v_i = v_i^2 = v_i^3 = \ldots$. This is the reason why there are no higher powers of the v_i in the definition of a monomial.
- The above also implies that the product of monomials is again a monomial, and the product of polynomials is a polynomial.
- There are 2^m monomials in the Boolean algebra on V^m (because there are 2^m subsets of $\{1, \ldots, m\}$).
- The weight of a monomial is calculated in the following result.

Lemma 11.1: weight of a monomial

A monomial $v_{i_1} v_{i_2} \ldots v_{i_r}$ in the Boolean algebra on V^m has weight 2^{m-r}. That is,

$$w(v) = 2^{m - \deg v} \quad \text{if } v \text{ is a monomial.}$$

Proof. If $b = x_1 x_2 \ldots x_m$ is a binary word, $v_{i_1} v_{i_2} \ldots v_{i_r}(b) = 1$ if and only if $x_{i_1} = x_{i_2} = \cdots = x_{i_r} = 1$. Hence the number of binary words in V^m where this monomial has value 1 is equal to the number of ways to choose the bits x_j where $j \notin \{i_1, \ldots, i_r\}$. There are 2 choices (0 or 1) for each one of those $m - r$ bits, hence the total number of such binary words is 2^{m-r}, and $w(v_{i_1} \ldots v_{i_r}) = \#\{b \in V^m : v_{i_1} \ldots v_{i_r}(b) = 1\} = 2^{m-r}$. $\square$

Theorem 11.2: monomial basis

Monomials form a basis of the Boolean algebra.

Proof. First, we prove by contradiction that monomials are *linearly independent*.

Assume for contradiction that a non-empty linear combination (i.e., a sum, as we are working over $\mathbb{F}_2$) of monomials equals the zero Boolean function:

$$v_{S_1} + v_{S_2} + \cdots + v_{S_k} = 0, \qquad k \geq 1,$$

where $S_1, \ldots S_k$ are some subsets of the index set $\{1, \ldots, m\}$. Without the loss of generality, assume that v_{S_k} has the highest degree:

$$\deg v_{S_i} \leq \deg v_{S_k}, \quad \text{i.e.,} \quad \#S_i \leq \#S_k \quad \text{for all } i = 1, \ldots, k-1.$$

Note that if $S, T \subseteq \{1, \dots, m\}$ then $v_S v_T = v_{S \cup T}$. Let now $T = \{1, \dots, m\} \setminus S_k$, the complement of the set S_k. Multiplying both sides by v_T, we obtain

$$v_{S_1 \cup T} + v_{S_2 \cup T} + \cdots + v_{S_k \cup T} = 0. \qquad (*)$$

We have $S_k \cup T = \{1, \dots, m\}$. If $i < k$ then the set S_i cannot contain S_k, and so $S_i \cup T \neq \{1, \dots, m\}$ and $\deg v_{S_i \cup T} < m$. Rewrite $(*)$ as

$$v_{S_1 \cup T} + v_{S_2 \cup T} + \cdots + v_{S_{k-1} \cup T} = v_1 v_2 \dots v_m.$$

The left-hand side is a sum of monomials of degree less than m. By Lemma 11.1, these monomials have value vectors of even weight. A sum of vectors of even weight is a vector of even weight: we know that the binary even weight code is linear. But the right-hand side is the monomial $v_1 \dots v_m$ which by Lemma 11.1 has weight 1, which is odd. This contradiction proves that monomials are linearly independent.

It remains to show that the monomials are a *spanning set* in the Boolean algebra. There are 2^m monomials, so we can form $2^{(2^m)}$ linear combinations of monomials by putting a coefficient of 0 or 1 in front of each monomial. All these linear combinations are distinct, by linear independence. On the other hand, there are $2^{(2^m)}$ Boolean functions on V^m. Hence every Boolean function is a linear combination of monomials.

A *basis* is a set which is linearly independent and spanning, so the Theorem is proved. $\square$

Corollary 11.3: Boolean functions are polynomials

Each Boolean function on V^m is uniquely written as a Boolean polynomial in the coordinate functions $v_1, \dots, v_m$.

Remark: algebraic normal form. A representation of a Boolean function $f\colon V^m \to \mathbb{F}_2$ as a Boolean polynomial is sometimes referred to as the *algebraic normal form* of f. This can be compared to *disjunctive* and *conjunctive* normal forms of a Boolean function used for other purposes. Interested readers may find the details in the literature.

The Reed-Muller code

We now know that every element of the Boolean algebra on V^m is a polynomial, i.e., a sum of several monomials (squarefree products of coordinate functions). Recall also that the degree of a polynomial is the top degree of a monomial in that polynomial, which does not exceed m.

Definition: Reed-Muller code

Let $0 \leq r \leq m$. The **rth order Reed-Muller code on V^m**, denoted $R(r, m)$, is the space of value vectors of polynomials of degree *at most* r in the Boolean algebra on V^m.

Observe that $R(r, m)$ is spanned by the value vectors of all monomials of degree at most r.

Example: work out $R(0, m)$

Find the parameters and write down all codevectors of the Reed-Muller code $R(0, m)$.

Solution. The code $R(0, m)$ consists of value vectors of Boolean polynomials on V^m of degree ≤ 0. There are only two such polynomials, $\mathbf{0}$ and $\mathbf{1}$, hence

$$R(0, m) = \{00\ldots0, 11\ldots1\} = \mathrm{Rep}(2^m, \mathbb{F}_2)$$

is the repetition code. The length is $2^m = \#V^m$. The dimension is 1. The minimum distance equals the length. A $[2^m, 1, 2^m]_2$-code.

Example: $R(m, m)$

Show that $R(m, m) = \mathbb{F}_2^{2^m}$, the trivial binary code of length 2^m.

Solution. $R(m, m)$ consists of value vectors of polynomials on V^m of degree $\leq m$. All Boolean polynomials have degree at most m, and, by Corollary 11.3, every possible binary vector of length 2^m is a value vector of some polynomial. Hence $R(m, m)$ consists of all possible binary vectors of length 2^m, i.e., is the trivial code.

The key result on Reed-Muller codes is the following theorem, which gives the parameters of these codes.

Theorem 11.4: parameters of a Reed-Muller code

$R(r, m)$ has length 2^m, dimension $\binom{m}{0} + \binom{m}{1} + \ldots + \binom{m}{r}$ and minimum distance 2^{m-r}.

Proof. **Length** $= 2^m$ by construction: a value vector is made up of 2^m bits obtained by evaluating the given function on the 2^m binary words in V^m.

Value vectors of monomials of degree $0, 1, \ldots, r$ span $R(r, m)$ by definition of $R(r, m)$, and are linearly independent by Theorem 11.2, hence form a basis of $R(r, m)$. The number of monomials of degree d is the same as the number of d-element subsets of $\{1, \ldots, m\}$, which is $\binom{m}{d}$, so the total number of monomials in the basis of $R(r, m)$ — i.e., the **dimension** of $R(r, m)$ — is as stated.

Minimum distance: the code $R(r,m)$ contains monomials of degree r, for example, $v_1v_2\dots v_r$. By Lemma 11.1, these have weight 2^{m-r}. Hence $d(R(r,m)) = w(R(r,m))$ is at most 2^{m-r}.

It remains to show that $w(R(r,m)) \ge 2^{m-r}$. We do this by **induction in** m**.**

Base case $m=1$. According to the Examples above, the two possible codes are $R(0,1) = \mathrm{Rep}(2,\mathbb{F}_2)$ of weight $2 = 2^{1-0}$ and $R(1,1) = \mathbb{F}_2^{2^m}$ of weight $1 = 2^{1-1}$. So the inequality $w(R(r,m)) \ge 2^{m-r}$ is satisfied when $m=1$.

Inductive step. Assume $w(R(r,m-1)) \ge 2^{m-1-r}$ for all $r = 0,\dots,m-1$. This means that the weight of any non-zero polynomial of degree $\le r$ in $v_1,\dots,v_{m-1}$ is at least 2^{m-1-r}:

$$h \neq 0,\ \deg h \le r \quad\implies\quad \#\{y \in V^{m-1} : h(y) = 1\} \ge 2^{m-1-r}. \tag{$\dagger$}$$

The set V^m of binary words of length m splits into two subsets,

$$V^{m-1}0 = \{x_1\dots x_m : x_m = 0\} \quad \text{and} \quad V^{m-1}1 = \{x_1\dots x_m : x_m = 1\}$$

of words that end in 0 and words that end in 1, respectively. We need to take a polynomial $0 \neq f\colon V^m \to \mathbb{F}_2$ of degree $\le r$ and prove that $w(f) \ge 2^{m-r}$. We have

$$w(f) = \#\{b \in V^{m-1}0 : f(b) = 1\} + \#\{b \in V^{m-1}1 : f(b) = 1\}. \tag{$\ddagger$}$$

Each monomial in f contains a copy of v_m or none, so we can write

$$f = g + hv_m,$$

where g, h are polynomials in $v_1,\dots,v_{m-1}$.

The case $h = 0$. Here g is a non-zero polynomial of degree $\le r$ in $v_1,\dots,v_{m-1}$, and so $r \le m-1$. By ($\dagger$), there are at least 2^{m-1-r} words $y \in V^{m-1}$ where $g(y) = 1$. For each such word y we have $y0 \in V^{m-1}0$, $y1 \in V^{m-1}1$ and $f(y0) = f(y1) = 1$, and so y contributes twice when counting the weight of f in ($\ddagger$). Hence $w(f) = 2w(g)$ and so $w(f) \ge 2 \times 2^{m-1-r} = 2^{m-r}$.

The case $h \neq 0$. We note that the values of f on $V^{m-1}0$ are the same as the values of g on V^{m-1}, because $hv_m|_{V^{m-1}0} = 0$. Furthermore, the values of f on $V^{m-1}1$ are the same as the values of $g+h$ on V^{m-1}, because on $V^{m-1}1$ we have $v_m = 1$. Hence ($\ddagger$) gives $w(f) = w(g) + w(g+h)$.

By the triangle inequality, $w(\underline{a}+\underline{b}) \le w(\underline{a}) + w(\underline{b})$ for any vectors $\underline{a}, \underline{b}$. Hence $w(g) + w(g+h) \ge w(g + (g+h)) = w(h)$. Here $\deg h \le r-1$ because $\deg hv_m \le r$, so the inductive hypothesis ($\dagger$) applies and gives $w(h) \ge 2^{m-1-(r-1)} = 2^{m-r}$. We proved that $w(f) \ge 2^{m-r}$, as required.

To conclude, by induction $w(R(r,m)) \ge 2^{m-r}$ for all m and all $r \le m$. $\square$

The key duality between Reed-Muller codes

We finish the chapter by identifying the dual code of $R(r, m)$, which happens to be another Reed-Muller code.

Theorem 11.5: duality between Reed-Muller codes

For all $m \geq 1$ and for all r such that $0 \leq r \leq m-1$,

$$R(m-1-r, m) = R(r, m)^{\perp}.$$

Proof. If $f, g \colon V^m \to \mathbb{F}_2$ are Boolean functions, the definition of inner product means that

$$\underline{f} \cdot \underline{g} = \sum_{b \in V^m} f(b)g(b) = \sum_{b \in V^m} (fg)(b).$$

If f is a monomial of degree $\leq r$ and g is a monomial of degree $\leq m-1-r$, then fg is a monomial of degree $\leq m-1$. By Lemma 11.1, there are exactly $2^{m-\deg fg}$ words $b \in V^m$ such that $(fg)(b) = 1$. Since $m - \deg fg \geq 1$, $2^{m-\deg fg}$ is an even number, and so the sum $\sum_{b \in V^m}(fg)(b)$ is zero in $\mathbb{F}_2$. This shows that f is orthogonal to g.

Since monomials f of degree $\leq r$ span $R(r, m)$, this shows that $g \in R(r, m)^{\perp}$. Thus, $R(m-1-r, m)$ is spanned by elements of $R(r, m)^{\perp}$, so $R(m-1-r, m) \subseteq R(r, m)^{\perp}$.

We will now compare the dimensions. We have $\dim R(m-1-r, m) = \binom{m}{0} + \cdots + \binom{m}{m-1-r}$. Using the relation $\binom{m}{i} = \binom{m}{m-i}$, we rewrite this as $\binom{m}{m} + \binom{m}{m-1} + \cdots + \binom{m}{r+1}$. Finally, $\dim R(m-1-r, m) + \dim R(r, m) = \sum_{i=0}^{m} \binom{m}{i} = 2^m$, the length of the Reed-Muller codes. Hence $\dim R(m-1-r, m) = 2^m - \dim R(r, m) = \dim R(r, m)^{\perp}$.

Thus, $R(r, m)^{\perp}$ contains subspace $R(m-1-r, m)$ of the same dimension as $R(r, m)^{\perp}$, hence a subset $R(m-1-r, m)$ of the same cardinality as $R(r, m)^{\perp}$. We conclude that $R(r, m)^{\perp} = R(m-1-r, m)$. □

Exercise. The code $R(m, m)$ is excluded from Theorem 11.5. How would you define "$R(-1, m)$" which should be the dual of $R(m, m)$?

Theorem 11.5 can be used to identify particular Reed-Muller codes and to deduce their further properties. Examples of this are in the exercises to this chapter.

Week 11

Exercises (answers at end)

Version 2023-12-03. To accessible online version of these exercises

Exercise 11.1 (identification of the Reed-Muller codes with $m = 3$)**.** Let $m = 3$. Write down the value vectors (in $\mathbb{F}_2^8$) of all the monomials in the Boolean algebra. Hence find generator matrices of the codes $R(r, 3)$, $0 \le r \le 3$. Try to recognise the codes obtained.

Partial answer. We use a slightly unconventional ordering of binary words in V^3. The value vectors of all the monomials in the Boolean algebra with $m = 3$:

	001	010	011	100	101	110	111	000
$\mathbf{1}$	1	1	1	1	1	1	1	1
v_1	0	0	0	1	1	1	1	0
v_2	0	1	1	0	0	1	1	0
v_3	1	0	1	0	1	0	1	0
v_1v_2	0	0	0	0	0	1	1	0
v_1v_3	0	0	0	0	1	0	1	0
v_2v_3	0	0	1	0	0	0	1	0
$v_1v_2v_3$	0	0	0	0	0	0	1	0

Exercise 11.2 ("the Mariner 9 code")**.** Check that $R(1, 5)$ is a $[32, 6, 16]_2$ code and detects up to 15 errors in a 32-bit codeword.

Exercise 11.3. Show that $R(r, m)$ is a self-orthogonal code, if and only if $r < m/2$.

Exercise 11.4. Show that the code $R(m-2, m)$ is, up to linear equivalence, an extended Hamming code $\widehat{\mathrm{Ham}}(m, 2)$.

Week 11

Exercises — solutions

Version 2023-12-03. To accessible online version of these exercises

Exercise 11.1 (identification of the Reed-Muller codes with $m = 3$)**.** Let $m = 3$. Write down the value vectors (in $\mathbb{F}_2^8$) of all the monomials in the Boolean algebra. Hence find generator matrices of the codes $R(r, 3)$, $0 \le r \le 3$. Try to recognise the codes obtained.

Partial answer. We use a slightly unconventional ordering of binary words in V^3. The value vectors of all the monomials in the Boolean algebra with $m = 3$:

	001	010	011	100	101	110	111	000
$\mathbf{1}$	1	1	1	1	1	1	1	1
v_1	0	0	0	1	1	1	1	0
v_2	0	1	1	0	0	1	1	0
v_3	1	0	1	0	1	0	1	0
v_1v_2	0	0	0	0	0	1	1	0
v_1v_3	0	0	0	0	1	0	1	0
v_2v_3	0	0	1	0	0	0	1	0
$v_1v_2v_3$	0	0	0	0	0	0	1	0

Answer to E11.1. A generator matrix for $R(0, 3)$ is formed by the value vector of 1, hence is $G_0 = [\,11111111\,]$. We conclude that

$$R(0, 3) = \mathrm{Rep}(8, \mathbb{F}_2).$$

The value vectors of 1, v_1, v_2, v_3 form a generator matrix G_1 of $R(1, 3)$. A really interesting code of length 8 and dimension 4. From the general theory of Reed-Muller codes we know that $R(r, m) = R(m-1-r, m)^{\perp}$. In particular, $R(1, 3)$ is a self-dual binary code. This can be checked directly: the rows of G_1 are of even weight (meaning that every row is orthogonal to itself) and are pairwise orthogonal.

Note that rows v_1, v_2, v_3 form a 3×8 matrix $\widehat{H}$ which contains each 3-bit column once. This is the generator matrix of the simplex code $\Sigma(3,2)$ with zero column appended — in other words, the extended simplex code $\widehat{\Sigma}(3,2)$.

Note that every vector $\underline{\hat{c}}$ in the extended Hamming code $\widehat{\mathrm{Ham}}(3,2)$ satisfies $\underline{\hat{c}}\widehat{H}^T = 000$, as the first 7 bits of $\underline{\hat{c}}$ form a Hamming codevector, and the last bit of $\underline{\hat{c}}$ is not used in $\underline{\hat{c}}\widehat{H}^T$.

Note also that $\underline{\hat{c}} \cdot 11111111 = 0$ as, by definition of an extended code, $\underline{\hat{c}}$ has even weight.

Hence $\underline{\hat{c}}G_1 = 0000$, and the matrix G_1, formed by rows $\mathbf{1}$, v_1, v_2, v_3, is a check matrix for $\widehat{\mathrm{Ham}}(3,2)$. But $\widehat{\mathrm{Ham}}(3,2) = \mathcal{H}_8$ is a self-dual code, as seen in earlier exercises. We conclude that G_1 is also a generator matrix for this code, hence

$$R(1,3) = \widehat{\mathrm{Ham}}(3,2).$$

The code $R(2,3)$ is generated by the top seven rows in the table above. We have

$$R(2,3) = R(0,3)^\perp = \mathrm{Rep}(8,\mathbb{F}_2)^\perp = E_8.$$

Finally, $R(3,3) = \mathbb{F}_2^8$ is the trivial binary code, of length 8 and dimension 8.

Exercise 11.2 ("the Mariner 9 code")**.** Check that $R(1,5)$ is a $[32,6,16]_2$ code and detects up to 15 errors in a 32-bit codeword.

Answer to E11.2. Put $r = 1$ and $m = 5$. The length of $R(1,5)$ is $2^5 = 32$ and $\dim R(1,5) = \binom{5}{0} + \binom{5}{1} = 1 + 5 = 6$. The minimum distance of $R(1,5)$ is $2^{5-1} = 16$. The code $R(1,5)$ is binary, as are all Reed-Muller codes. Hence it is a $[32,6,16]_2$-code as claimed.

Trivia: The code $R(1,5)$ was used by NASA Mariner 9 space probe to transmit greyscale images of the surface of Mars to Earth in 1972. It is a $[32,6,16]_2$ code. Each pixel was a 6-bit message, representing 64 grey values, and encoded as a 32-bit codeword. The code corrected up to 7 errors in a codeword (*wasn't that an overkill?..*)

Exercise 11.3. Show that $R(r,m)$ is a self-orthogonal code, if and only if $r < m/2$.

Answer to E11.3. $R(r,m)$ is self-orthogonal $\iff R(r,m) \subseteq R(r,m)^\perp = R(m-1-r,m)$ $\iff r \le m-1-r \iff 2r \le m-1 \iff 2r < m \iff r < m/2$.

Exercise 11.4. Show that the code $R(m-2,m)$ is, up to linear equivalence, an extended Hamming code $\widehat{\mathrm{Ham}}(m,2)$.

Answer to E11.4. *Sketch of proof.* Order the binary words in V^m so that the zero word $00\ldots0$ comes last.

The value vectors of $v_1,\ldots,v_m$ form a matrix whose last column is zero, preceded by $2^m - 1$ distinct non-zero m-bit columns. This is the generator matrix of $\widehat{\Sigma}(m,2)$.

Hence any $\underline{\hat{c}} \in \widehat{\mathrm{Ham}}(m,2)$ is orthogonal to rows $v_1, \ldots, v_m$. Note that $\underline{\hat{c}}$ has even weight, hence is orthogonal to row $\mathbf{1}$ which consists of all ones.

This shows that $\widehat{\mathrm{Ham}}(m,2)$ lies inside the dual code to the code spanned by $\mathbf{1}, v_1, \ldots, v_m$. That is, $\widehat{\mathrm{Ham}}(m,2) \subseteq R(1,m)^\perp$. The dimension of both sides is $2^m - (m+1)$, so we conclude that $\widehat{\mathrm{Ham}}(m,2) = R(1,m)^\perp$. It remains to note that $R(1,m)^\perp = R(m-2,m)$ by a result from the course.