IMPLEMENTATION AND COMPARISON OF THE GOLAY AND FIRST ORDER REED-MULLER CODES

by

Olga Shukina

A Thesis Submitted to the Faculty of

The Charles E. Schmidt College of Science

in Partial Fulfillment of the Requirements for the Degree of

Master of Science

Florida Atlantic University

Boca Raton, FL

August 2013

IMPLEMENTATION AND COMPARISON OF THE GOLAY AND FIRST ORDER REED-MULLER CODES

by

Olga Shukina

This thesis was prepared under the direction of the candidate's thesis advisor, Dr. Spyros Magliveras, Department of Mathematical Sciences, and has been approved by the members of her supervisory committee. It was submitted to the faculty of the Charles E. Schmidt College of Science and was accepted in partial fulfillment of the requirements for the degree of Master of Science.

SUPERVISORY COMMITTEE:

Spyros Magliveras, Ph.D.
Thesis Advisor

Lee Klingler, Ph.D.

Warren McGovern, Ph.D.

Lee Klingler, Ph.D.
Chair, Department of Mathematical Sciences

Ingrid B. Johanson, Ph.D.
Dean, The Charles E. Schmidt College of Science

Barry T. Rosson, Ph.D.
Dean, Graduate College

7/10/2013
Date

ACKNOWLEDGEMENTS

I would like to express the deepest appreciation and sincere gratitude to my advisor, Dr. Spyros Magliveras, for the continuous support, patience, motivation, enthusiasm, and immense knowledge. I could not have imagined having a better advisor and mentor. Without his guidance and persistent help this thesis would not have been possible.

I would also like to thank the rest of my thesis committee, Dr. Lee Klingler and Dr. Warren McGovern. I had the pleasure of attending their classes. Prof. Magliveras, Prof. Klingler and Prof. McGovern are the most influential and life-changing role models and teachers. I have learned through them, through their commitment to excellence.

I also take this opportunity to express my sincere thanks to all the faculty members of the Department of Mathematics for their help and encouragement.

Last but not the least, I would like to thank my family: mom and dad, husband James and my little Daniel.

ABSTRACT

Author:	Olga Shukina
Title:	Implementation and Comparison of the Golay and First Order Reed-Muller codes
Institution:	Florida Atlantic University
Thesis Advisor:	Dr. Spyros Magliveras
Degree:	Master of Science
Year:	2013

In this project we perform data transmission across noisy channels and recover the message first by using the Golay code, and then by using the first-order Reed-Muller code. The main objective of this thesis is to determine which code among the above two is more efficient for text message transmission by applying the two codes to exactly the same data with the same channel error bit probabilities. We use the comparison of the error-correcting capability and the practical speed of the Golay code and the first-order Reed-Muller code to meet our goal.

DEDICATION

To James Meredith.

IMPLEMENTATION AND COMPARISON OF THE GOLAY AND FIRST ORDER REED-MULLER CODES

LIST OF FIGURES

Chapter 1

Background

The subject of error-correcting codes arose in response to practical problems in the reliable communication of digitally encoded information. It started in 1948, when Claude Shannon gave a formal description of a communication system. He also introduced a beautiful theory about the concept of information. Claude Shannon's paper "A Mathematical Theory of Communication" marked the birth of a new subject called "Information Theory", part of which is coding theory. C. Shannon established the theoretical foundation of the subject. He showed that "good codes" exist without exhibiting them. His proof was probabilistic and existential but not constructive. It remained a big challenge to construct and implement efficient codes for a very long time. While Claude Shannon was developing information theory and coding as a mathematical model for communication in the late 1940's, his colleague at Bell Laboratories Richard Hamming found a need for error correction in his own work on computers. Setting to work on this problem R. Hamming created a way of encoding information so that if an error was detected it could also be corrected. Richard Hamming was one of the first to actually construct and implement error correcting codes. Inspired by Hamming's work, Claude Shannon developed the theoretical framework for the science of coding theory. The theory of error detecting and correcting codes became a branch of mathematics and engineering which deals with the reliable trans-

mission and storage of data. Coding, and in particular error-control coding, is an extremely important part of applied mathematics. The sharing and the transmission of data are an integral part of communication in the world. Coding makes data transmission much easier by putting the data into a simpler form. Error-control codes help prevent the miscommunication of a message by working to correct any mistakes or scrambling of the message that occurs during transmission. Error-control codes are used to detect and correct errors that occur when data are transmitted across some noisy channel or stored on some medium. This theory has been developed for such diverse applications as the minimization of noise from compact discs and digital versatile disc recordings, the transfer of financial information across telephone lines and data transfer from one computer to another computer. When photographs are transmitted to earth from space, error-control codes are used to protect from any noise caused by different atmospheric interference. Compact discs use error-control codes so that a playing device reads data from a compact disc even if it has been corrupted by noise in the form of imperfections on a disc. Today, coding is used in a broad range of communication systems. It has become increasingly important in the development of new technologies for data communications and data storage.

Chapter 2

Introductory Concepts

2.1 MODEL OF A DIGITAL COMMUNICATIONS SYSTEM

Information media are not very reliable in real life since data get distorted by many forms of interference collectively called *noise*. We can deal with this undesirable situation by incorporating some form of redundancy in the original data. With this redundancy the original data can be recovered, or at least the errors can be detected, if errors are introduced up to some tolerance level. The following two examples will illustrate this concept.

Example 2.1.1. The usual way to represent, manipulate and transmit information is to use bit strings, that is, sequences of zeros and ones. A simple way to detect errors when a bit string is transmitted is to add a *parity check bit* at the end of the string. If the bit string contains an even number of ones we put 0 at the end of the string. If the bit string contains an odd number of ones we put 1 at the end of the string. Adding the parity check bit guarantees that the number of ones is even.

Suppose that a parity check bit is added to a bit string before it is transmitted. If the receiver reads 1110011 it concludes that errors have occurred since 1110011 contains an odd number of ones. If the receiver reads 10111101 it concludes either no error has occurred or an even number of errors have occurred.

The example above demonstrates that transmission error detection is possible by

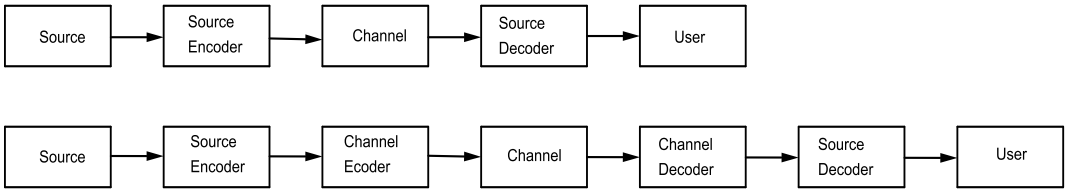


Figure 2.1: Communications Channel

introducing redundancy. We can do even better if we include more redundancy.

Example.2.1.2. Suppose we want to transmit the information that comes from the set of symbols $\{A, B, C, D\}$. We associate sequences of zeros and ones with each of these symbols.

$A \to 00$

$B \to 10$

$C \to 01$

$D \to 11$

The receiver will interpret a 00 sequence as the message A, a 10 sequence as the message B and so on. This process can be represented by the simple diagram of the communications channel. (Figure 2.1)

The *source* emits symbols from the message set. The *source encoder* associates each symbol with a binary sequence and then transmits it. We think of a *channel* as any information medium. The *source decoder* receives the binary sequence, converts it back to the symbols and passes it to the *user* as a message. There is a certain probability that what the decoder receives is not what was sent due to unreliability of a channel. We conclude that a single error has occurred if the source encoder sends 00 and the source decoder receives 10. 10 is a valid piece of information and the decoder has no way of knowing which single error has occurred. The decoder will pass the symbol B to the user. We can improve the reliability of message transmission

by adding redundancy to each message. The redundancy will help us detect and possibly correct channel errors. In our example we choose to add the redundancy in the following way:

$A \to 00 \to 00000$

$B \to 10 \to 10110$

$C \to 01 \to 01011$

$D \to 11 \to 11101$

If the receiver reads 01000, it is obvious that an error has occurred. It is reasonable for the decoder to assume the transmitted sequence was 00000 since the received sequence can be obtained from 00000 by the introduction of only one error. At least two errors would have to occur to obtain 01000 from any of the other three sequences. Altering a single bit in any one of the above 5-bit sequences will result in a unique sequence. Therefore if a single bit is altered, the resulting sequence can be easily identified with one of the original sequences.

This example illustrates the new coding process which enable us to correct a single error. This new coding process is described by the following diagram: (Figure 2)

Definition 2.1.1. *Let A be an alphabet of q symbols. A block code C of length n containing M codewords over A is a set of M n-tuples where each n-tuple takes its components from A. A code of length n and size M is called an (n, M)-code.*

The *binary alphabet* $A = \{0, 1\}$ will be used throughout this thesis. A code over $A = \{0, 1\}$ is called a *binary code*.

The source encoder transforms messages into k-tuples over the code alphabet A. The *channel encoder* assigns a codeword of length n to each of these k-tuples. We observe the message expansion since the channel encoder is adding redundancy ($n > k$). While adding redundancy is desirable for error control, it decreases the

efficiency of the communication channel. The redundancy is frequently expressed in terms of the *code rate*.

Definition 2.1.2. *The rate of an (n, M)-code which encodes information k-tuples is*

$$R = \frac{k}{n}.$$

The quantity $r = n - k$ is the redundancy.

Suppose a message $u = u_1...u_k$ is encoded into the codeword $x = x_1...x_n$, which is sent through the channel. The receiving end receives the sequence of length n, $y = y_1...y_n$, which is called the *received vector*. The *channel decoder* estimates from the received vector y the transmitted vector x and delivers the estimated codeword to the source decoder. The source decoder converts the received sequence of zeroes and ones into the message. If the channel is free of noise, the received vector y is equal to the transmitted codeword x. But if the channel is noisy the received vector y may be different from x. In this case the received vector is expressed as $y = x + e$, where vector $e = (e_1, e_2, ..., e_n)$ is the result of the channel noise. Vector e is called the *error vector*.

Example 2.1.3. Suppose the codeword $x = (01001)$ is transmitted.

If the errors occurred on the second and third bits of the codeword, the error vector is $e = (01100)$, and the received vector becomes $y = x + e = (00101)$.

Note that the above is ordinary sum of vectors (bitwise exclusive or) over the field $\mathbb{F}_2$.

2.2 HAMMING DISTANCE, HAMMING WEIGHT AND MINIMUM DISTANCE

One of the most important parameters associated with an (n, M)-code C is the *Hamming distance*

Definition 2.2.1. *Let $x = x_1, ..., x_n$ and $y = y_1, ..., y_n$. Then, for every i define*

$$d(x_i, y_i) = \begin{cases} 1 & x_i \neq y_i \\ 0 & x_i = y_i \end{cases}$$

and define

$$d(x, y) = \sum_{i=1}^{n} d(x_i, y_i)$$

The *Hamming distance $d(x, y)$ between two codewords x* and y is the number of coordinate positions in which they differ.

Example 2.2.1. The symbols of two vectors $x = (01110)$ and $y = (10110)$ differ in two positions. Therefore,

$$d(x, y) = 2.$$

Proposition 2.2.1. *(See [1]) The function d is a metric. That is, for every $x, y, z \in A^n$*

1. $0 \leq d(x, y) \leq n$
2. $d(x, y) = 0$ *if and only if* $x = y$
3. $d(x, y) = d(y, x)$
4. $d(x, z) \leq d(x, y) + d(y, z)$ *(triangle inequality)*

Definition 2.2.2. *The Hamming weight of a vector $x = x_1...x_n$ is the number of nonzero x_i, $1 \leq i \leq n$, and is denoted by wt(x).*

Example 2.2.2. The number of nonzero symbols in $x = (00111)$ is three. Therefore,

$$wt(x) = 3.$$

Definition 2.2.3. *Let C be an (n, M)-code. The minimum distance d of the code C is $d = min\ d(x, y) = min\ wt(x - y), x, y \in C, x \neq y$.*

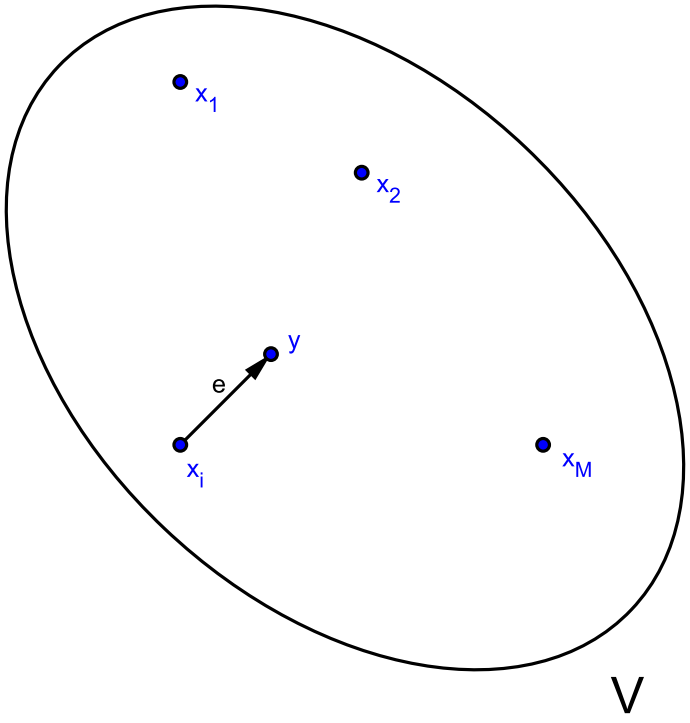
In other words, the minimum distance of a code is the minimum distance between two distinct codewords, over all pairs of codewords. The minimum distance of a code is an important parameter to describe the error-detection and error-correction capabilities of the code.

2.3 PRINCIPLES OF ERROR DETECTION

We define a set called *signal space* in order to illustrate the principles of error detection and error correction geometrically. The signal space is defined to be the set of all binary vectors of length n and is denoted by V.

A code is a subset of a signal space. A codeword of a code is represented by a point in the signal space. Thus, a code $C = \{x_1, x_2, ..., x_M\}$ of size M is represented by a set of M points in a signal space. A received vector is also a point in the signal space. Suppose a codeword x_i is transmitted and y is a received vector. The error $e = y - x_i$ is represented by a vector from x_i to y in the signal space.

First we consider the case where the code is used only for error detection. The decoder decides whether or not any error occurred in the transmission of a codeword. If $y \in C$, the decoder decides that there was no error on the channel and y is a transmitted codeword. Then the message block of k bits corresponding to y is delivered to the destination. If $y \notin C$, the decoder decides an error occurred in the channel. In this situation, the receiving end may send a request to the sender for a retransmission of the codeword. If the retransmission is impossible, the receiving end may estimate the message from the codewords received before and after the erroneous codeword. If y is equal to another codeword x_j $(j \neq i)$, the decoder makes a false decision that



Codewords and a received vector in the signal space.

Figure 2.2: Signal Space

there was no error and that the transmitted codeword was x_j. This type of error is called an *undetected error*.

A code is a subset of a signal space. A codeword of a code is represented by a point in the signal space. Thus, a code $C = \{x_1, x_2, ..., x_M\}$ of size M is represented by a set of M points in a signal space. A received vector is also a point in the signal space. Suppose a codeword x_i is transmitted and y is a received vector. The error $e = y - x_i$ is represented by a vector from x_i to y in the signal space.

First we consider the case where the code is used only for error detection. The decoder decides whether or not any error occurred in the transmission of a codeword.

If $y \in C$, the decoder decides that there was no error on the channel and y is a transmitted codeword. Then the message block of k bits corresponding to y is delivered to the destination. If $y \notin C$, the decoder decides an error occurred in the channel. In this situation, the receiving end may send a request to the sender for a retransmission of the codeword. If the retransmission is impossible, the receiving end may estimate the message from the codewords received before and after the erroneous codeword. If y is equal to another codeword x_j ($j \neq i$), the decoder makes a false decision that there was no error and that the transmitted codeword was x_j. This type of error is called an *undetected error*.

If the minimum distance of a code C is $d = 2t + 1$, t a positive integer, then spheres of radius t with centers the codewords do not intersect, so C corrects up to t errors. If $d = 2t$ then spheres of radius $t - 1$ are disjoint, but spheres of radius t may intersect. In the latter case, C corrects up to $t - 1$ errors but will also detect if t errors have occurred.

2.4 PRINCIPLES OF ERROR CORRECTION

We divide the signal space into regions $S_r(x_1), S_r(x_2), ..., S_r(x_M)$, called *spheres*, in order to perform error correction at the decoder. These regions correspond to $x_1, x_2, ..., x_M$ respectively.

Definition 2.4.1. *A sphere of radius r about a vector u, denoted by $S_r(u)$, is the set of all vectors in the space whose distance from u is less than or equal to r.*

$$S_r(u) = \{v \in V \mid d(u, v) \leq r\}.$$

The decoder decides that the codeword x_i was transmitted when the received vector y is in the sphere $S_r(x_i)$.

Suppose the codeword x_i is transmitted. If an error e occurs, so that the received vector $y = x_i + e$ is in the sphere $S_r(x_i)$, the error is corrected correctly by the decoder.

If an error occurs so that the received vector is contained in sphere $S_r(x_j)$ $i \neq j$, the decoder decides x_j was transmitted. The decoder commits an error. This type of error is called a *decoding error*. The error correction capability of a code is determined by the minimum distance d.

Theorem 2.4.1. *If d is the minimum distance of a code C, then C can correct $t = \lfloor (d-1)/2 \rfloor$ or fewer errors.*

Proof: We want to show that spheres of radius $t = \lfloor (d-1)/2 \rfloor$ about codewords are disjoint. Suppose not. Let u and v be distinct vectors in code C. Assume that $S_t(u) \cap S_t(v)$ is nonempty. Suppose $w \in S_t(u) \cap S_t(v)$. Then $d(u,v) \leq d(u,w) + d(w,v) \leq 2t$ by the triangle inequality and because the points are in the spheres. Now $2t \leq d-1$ so that $d(u,v) \leq d-1$. But $d(u,v)$ must be greater or equal to d since u and v are codewords and the minimum distance of a code is d. This contradiction shows that the spheres of radius t about codewords are disjoint. This means that if t or fewer errors occur, the received vector is in a sphere of radius t about unique codeword. □

In the decoding procedure the decoder has to decide which codeword was transmitted. When the decoder receives vector y it must make one of the three possible decisions:

1. no errors have occurred, accept y as a codeword;

2. errors have occurred, correct y;

3. errors have occurred, correction is impossible.

The decoder will not always make a correct decision. For example, consider the possibility of an error pattern occurring which changes a transmitted codeword into another codeword. It is desirable to minimize the probability of a decoding error to reduce damage at the destination. The strategy that minimizes the probability of the decoder making a mistake is called *maximum likelihood decoding*, provided the codewords are equally likely. This strategy will enable the decoder to choose the *most likely* error vector e. Maximum likelihood decoding is obtained by choosing the nearest codeword from the received vector. Such a decoding method is called *nearest neighbor decoding*.

One of the simplest possible type of communications channel is a *binary symmetric channel*. It has no memory and it receives and transmits only two symbols, 0 and 1. A binary symmetric channel has two probabilities:

1. $Pr[1 \text{ received}|0 \text{ was sent}] = Pr[0 \text{ received}|1 \text{ was sent}] = p$ (probability that an error occurs on symbol transmission)

2. $Pr[1 \text{ received}|1 \text{ was sent}] = Pr[0 \text{ received}|0 \text{ was sent}] = 1 - p$ (probability that a symbol is correctly transmitted over the channel)

The probability p is called the *crossover probability*.

Theorem 2.4.2. *In a binary symmetric channel with crossover probability $p < \frac{1}{2}$, maximum likelihood decoding is equivalent to nearest neighbor decoding.*

Proof: Let C be a code and y the received word. Then for every x and for every i we have that $d(y, x) = i$ if and only if

$$Pr[y \text{ received}|x \text{ was sent}] = p^i(1-p)^{n-i}.$$

Since $p < \frac{1}{2}$ we have that $1 - p > p$ and $\frac{1-p}{p} > 1$. Thus

$$p^i(1-p)^{n-i} = p^{i+1}(1-p)^{n-i-1} \times \frac{1-p}{p} > p^{i+1}(1-p)^{n-i-1}.$$

This implies that

$$p^0(1-p)^n > p(1-p)^{n-1} > ... > p^n(1-p)^0$$

and so the nearest neighbor yields the codeword that maximizes the required probability. □

Example 2.4.1. Consider the binary code $C = \{(00000), (10110), (01011), (11101)\}$ and suppose that the symbol error probability for the channel is $p = 0.1$. If $y = (11111)$ is a received vector, then

$$Pr[y, (00000)] = Pr[e = (11111)] = (0.1)^5 = 0.00001,$$

$$Pr[y, (10110)] = Pr[e = (01001)] = (0.1)^2(0.9)^3 = 0.00729,$$

$$Pr[y, (01011)] = Pr[e = (10100)] = (0.1)^2(0.9)^3 = 0.00729,$$

$$Pr[y, (11101)] = Pr[e = (00010)] = (0.1)^1(0.9)^4 = 0.06561.$$

Since the probability $Pr[y, (11101)] = 0.06561$ is largest, y is decoded to (11101).

The probability that a codeword x sent over the channel is correctly decoded at the receiving end shows the reliability of nearest neighbor decoding. Suppose the channel introduces a symbol error with probability p. The codeword x will be correctly decoded if the decoder receives any vector in the sphere of radius $e = \lfloor \frac{(d-1)}{2} \rfloor$ about x. The probability of this is

$$\sum_{y \in S(x)} P(y, x) = \sum_{i=0}^{e} \binom{n}{i} p^i (1-p)^{n-i},$$

giving a lower bound on the probability that a transmitted codeword is correctly decoded. However, if the channel introduces too many errors in a single word, the decoder will decode incorrectly.

Example 2.4.2. Consider the code from example 2.4.1. The decoder will certainly decode correctly if the channel introduces at most one error to any codeword

since the minimum distance of the code is three ($d = 3$). If the probability of a symbol error $p = 0.1$ the probability that the nearest neighbor decoding applied to the code decodes a transmitted codeword correctly is

$$(0.9)^5 + \binom{5}{1}(0.1)^1(0.9)^4 = 0.91854.$$

Suppose that the channel error rate changes so that $p = 0.4$. The probability then becomes

$$(0.6)^5 + \binom{5}{1}(0.4)^1(0.6)^4 = 0.33696.$$

The nearest neighbor decoding becomes less reliable as the symbol error rate increases. This brings up an important point in the design of a code. A code should be designed appropriately depending on the expected rate of errors for the channel being employed.

2.5 AUTOMORPHISMS

Although in this thesis we do not use automorphisms of codes in any implementation aspects, we would still like to define automorphisms of codes because of their importance in characterizing the [24,12,0] binary linear Golay code which we use later.

It is often convenient to display an $(n, M)-$code C code as an $M \times n$ matrix A_C, whose rows are the codewords of C. For example, the code $C = \{(00000), (01010), (11001), (10110), (01011), (11101)\}$ corresponds to the matrix :

$$A_C = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix}$$

Definition 2.5.1. *Two $(n, M)-$codes C_1 and C_2 are said to be equivalent, denoted by $C_1 \sim C_2$, if C_2 can be obtained from C_1 by applying a fixed permutation to the coordinates of each of the codewords of C_1.*

For example, permuting the codeword coordinates of $C_1 = \{(00000), (01010), (11001), (10110), (01011), (11101)\}$, according to permutation $\pi = (1\ 2\ 3)(4\ 5)$ yields code $C_2 = \{(00000), (00101), (01110), (11001), (00111), (11110)\}$. Thus, $C_1 \sim C_2$. Permuting the codeword coordinates of C_1 by applying π is best visualized by looking at the corresponding matrices. The matrix A_{C_2} is simply the result of applying π to the columns of A_{C_1}.

$$A_{C_1} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \quad \xrightarrow{\pi} \quad A_{C_2} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$$

Of course, if the rows of A_C are permuted, then we get a new matrix representing the same code C: the codewords are simply displayed in a different order.

Recall that permuting the columns of a matrix A according to permutation π corresponds to multiplying A (on the right) by the permutation matrix P_π corresponding

to π. Thus

$$A_{C_1} \cdot P_\pi = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix} = A_{C_2}$$

Definition 2.5.2. *Let C be an (n, M) code. Then a permutation $\pi \in \mathcal{S}_n$ is said to be an automorphism of C if and only if applying π to (the codeword coordinates of C) yields C.*

In terms of matrices, π is an automorphism of C if and only if $A_C \cdot P_\pi$ yields the same code, that is A_C possibly with its rows permuted. Thus, π is an automorphism if and only if there is an $M \times M$ permutation matrix Q such that:

$$A_C \cdot P_\pi = Q \cdot A_C$$

Example 2.5.1. *The permutation $\pi = (1\ 2\ 3\ 4)$ is an automorphism of the code*

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

The collection of all automorphisms of a given code C is a subgroup of the symmetric group $\mathcal{S}_n$ called the *(full) automorphism group* of C. Automorphisms of codes play an important role in efficiently decoding certain important codes, but we will not discuss these matters here.

Chapter 3

Linear Codes

3.1 BASIC DEFINITIONS

Almost all of the codes used in practice are *linear codes*. An algebraic structure of linear codes provides a framework for constructing encoding and decoding algorithms.

We denote a *finite field* with q elements by $\mathbb{F}_q$. The smallest finite field is $\mathbb{F}_2$, which consists of 0 and 1. $\mathbb{F}_2$ is also the filed most commonly used for communications and error correcting codes. Addition and multiplication in $\mathbb{F}_2$ are carried out as ordinary operations in $\mathbb{Z}$ modulo 2.

Suppose that the set of all messages we want to transmit is the set of k-tuples with components from some field $\mathbb{F}_q$. There are q^k messages. This set is a *vector space* $\mathbb{F}_q^k$ called a *message space*. The encoder generates the codewords by *embedding* the message k-tuples into n-tuples with $n \geq k$ for error detection and error correction. Thus, we set up a one-to-one correspondence between the q^k messages and q^k n-tuples in vector space $\mathbb{F}_q^n$. We add the redundancy so that q^k n-tuples form a k-dimensional subspace in $\mathbb{F}_q^n$. The number of codewords is q^k. The number of symbols n in a codeword is called the *length of a code*. The number of message symbols k is called the *dimension of a code*. An $[n, k]$-code is a linear code of length n and dimension k.

Definition 3.1.1. *A linear $[n, k]$-code C is a subspace of dimension k in the n-dimensional linear space over $\mathbb{F}_q$ which consists of all vectors of length n over $\mathbb{F}_q$.*

Since a linear code C is a vector space, it can be given by a basis. C is completely determined by k linearly independent vectors in $\mathbb{F}_q^n$. We can set up a one-to-one correspondence between the k-dimensional message space and the code C once a basis is selected. When a linear code is employed, the encoder generates the codeword $x = (x_1\, x_2\, ...\, x_n)$ from the message $u = (u_1\, u_2\, ...\, u_k)$ by the linear mapping

$$x = uG, \tag{3.1}$$

where $x_i, u_j \in \mathbb{F}_q$ and G is a $k \times n$ matrix with elements from $\mathbb{F}_q$. The matrix G is called a *generator matrix* of the code.

Definition 3.1.2. *A matrix G whose rows constitute a basis for linear code C is called a generator matrix for C.*

The rank of G is k since the k rows of G are linearly independent.

The codewords of a linear code C with generator matrix G are all linear combinations of the rows of G. We say that C is the code generated by the matrix G. A linear code C has many different generator matrices since subspace C has many different bases. In particular if G_1 and G_2 are two generator matrices for C if and only if they are related by a $k \times k$ non-singular matrix over $\mathbb{F}_q$, i.e. $G_1 = MG_2$.

Example 3.1.1 Consider the message space $M = \{(00), (10), (01), (11)\}$ over $\mathbb{F}_2$.
If we select the generator matrix $G_1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{pmatrix}$, we will obtain the code $C_1 = \{(00000), (10000), (01000), (11000)\}$.
If we select the generator matrix $G_2 = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$, we will obtain the code $C_2 = \{(00000), (10111), (11110), (01001)\}$.
If we select the generator matrix $G_3 = \begin{pmatrix} 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 \end{pmatrix}$, we will obtain the code $C_3 = \{(00000), (10110), (01011), (11101)\}$.

The example 3.1.1. demonstrates how one generator matrix may be more useful

than another one. The length of a code n determines the potential for error correction of the code. Hence, choosing n involves a tradeoff between the error-correcting capability and the rate of the code. Once k and n are both fixed, the codes may not have the same minimum distance depending on the selected basis. We want to choose a basis that provides the greatest minimum distance. In example 3.1.1. the minimum distance of the code C_1 is 1, the minimum distance of C_2 is 2, and the minimum distance of C_3 is 3. We conclude that C_3 is the "best code" among these three.

A linear code of length n, dimension k, and minimum distance d will be called an $[n, k, d]$-code.

It is not necessary to compare every pair of codewords to find the minimum distance of a linear code. By the definition 2.2.3. the minimum distance d of the code C is :

$$d = \min_{x,y \in C, x \neq y} d(x, y) = \min_{x,y \in C, x \neq y} wt(x - y)$$

If x and y belong to a linear code C, $x - y = z$ is also a codeword. Therefore, the minimum distance d of a linear code C is equal to the *Hamming weight of a code.*

Definition 3.1.3. *The Hamming weight of an $[n, k]$-code C is*

$$w(C) = min\{w(x) : x \in C, x \neq 0\}.$$

Theorem 3.1.1. *Let d be the minimum distance in an $[n, k]$-code C. Then $d = w(C)$.*

There are advantages in selecting one basis over another as we have seen before. A generator matrix G for a linear code is called *systematic* if it is of the form

$$G = [I_k \; A],$$

where I_k is $k \times k$ identity matrix and A is a $k \times (n - k)$ matrix. Such a matrix is said to be in *standard form.* When a systematic code is employed, the first k symbols of

the codeword are equal to the message symbols:

$$x_1 = u_1,\ x_2 = u_2,\ ...,\ x_k = u_k.$$

The first k symbols of the codeword are called the *message symbols* or *information symbols*, and the last $n - k$ symbols are called the *check symbols*.

Example 3.1.2. Suppose that the vectors (10000), (01010), and (00111) form a basis for a $[5, 3]$-code C over $\mathbb{F}_2$. Thus, the generator matrix is

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

Suppose the message space consists of binary 3-tuples. There are 8 messages in the message space. We will map a few messages to the codewords by using G. (010) gets mapped to (01010), (101) gets mapped to (10111), and (111) gets mapped to (11101). In each codeword the first three positions are precisely the information symbols or message itself. Therefore, the decoding is very simple. We need to take the first three components of the codeword to retrieve the message.

Since the rank of a generator matrix is k, any generator matrix can be transformed into a generator matrix in standard form by performing *elementary row operations* plus possibly column permutations. The set of codewords generated by a transformed generator matrix becomes identical to the set of codewords generated by a matrix before the transformation if the order of the symbols of the codewords is appropriately changed. Recall from section 2.5 that two codes C_1 and C_2 are *equivalent* if the sets of codewords of C_1 are transformed into the set of codewords of C_2 by applying a particular permutation π to the coordinates of the codewords. The following proposition is easy to deduce.

Proposition 3.1.1. *Two $[n, k]$-codes C_1 and C_2 over a field $\mathbb{F}_q$ are equivalent codes if there exist generator matrices G_1 and G_2 for C_1 and C_2 respectively and an $n \times n$ permutation matrix P such that*

$$G_2 = G_1 P.$$

Note that a permutation matrix is an identity matrix with rows or columns permuted.

The matrix P permutes the columns of G_1, and thus permutes the coordinate positions in C_1 to produce the code C_2.

Any linear code is equivalent to a systematic code. Equivalent codes have the same error-correcting properties since they have the same Hamming weight and distance.

We can also define a linear $[n, k]$-code C over a field $\mathbb{F}_q$ in terms of another subspace, called the *orthogonal complement* of C.

Definition 3.1.4. *Let $x = (x_1\ x_2\ ...\ x_n)$ and $y = (y_1\ y_2\ ...\ y_n)$ be vectors in $\mathbb{F}_q^n$. The inner product of x and y is*

$$x \cdot y = \sum_{i=1}^{n} x_i\, y_i,$$

where the sum is computed over the field $\mathbb{F}_q$. The vectors x and y are orthogonal to each other if $x \cdot y = 0$.

Definition 3.1.5. *Let C be an $[n, k]$-code over a field $\mathbb{F}_q$. The orthogonal complement $C^{\perp}$ of C is*

$$C^{\perp} = \{x \in \mathbb{F}_q^n :\ x \cdot y = 0\ for\ all\ y \in C\}.$$

Therefore, $C^{\perp}$ is the set of n-tuples over $\mathbb{F}_q$ that are orthogonal to every vector in C.

The orthogonal complement of C is a subspace and thus another linear code called the *dual code* of C. If C is $[n, k]$-code, then $C^{\perp}$ is an $[n, n-k]$-code.

Since vectors may be self-orthogonal, it is possible that $C = C^{\perp}$. A code for which $C = C^{\perp}$ is called a *self dual code.*

Suppose C is an $[n, k]$-code with a $k \times n$ generator matrix G. Since $C^{\perp}$ is an $[n, n-k]$-code, and hence must have an $(n-k) \times n$ generator matrix H. Since every vector in the row space of G is orthogonal to every vector in the row space of H, we must have $GH^T = 0$.

Definition 3.1.6. *Let C be an $[n, k]$-code over $\mathbb{F}_q$. If H is a generator matrix for $C^{\perp}$, then H is called a parity-check matrix for C.*

Theorem 3.1.2. *Let H be a parity-check matrix for an $[n, k]$-linear code C over $\mathbb{F}_q$. Then*

$$C = \{x \in \mathbb{F}_q^n : Hx^T = 0\},$$

where T denotes the transpose. In other words, a vector $x \in \mathbb{F}_q^n$ is a codeword of C if and only if

$$Hx^T = 0. \tag{3.2}$$

Proof: For any vector $x \in \mathbb{F}_q^n$, the quantity $s = Hx^T$ is an $(n-k)$-tuple whose components are the inner products of x with each of the $n-k$ basis vectors of $C^{\perp}$. If $s = 0$, then x is orthogonal to every such basis vector and hence by the linearity of $C^{\perp}$ is orthogonal to every codeword of $C^{\perp}$. Thus x must be an element of $(C^{\perp})^{\perp}$. Therefore $x \in C$.

If $x \in C$, then $Hx^T = 0$ since x is orthogonal to every vector of $C^{\perp}$. Thus $x \in C$ if and only if $Hx^T = 0$. □

Let $G = [I_k \; A]$ be a generator matrix of a code C in standard form, where A is a $k \times (n-k)$ matrix. G has rank k. Set

$$H = [-A^T \; I_{n-k}], \tag{3.3}$$

then H has rank $n-k$. Since $GH^T = A - A = 0$, it follows that H is a parity check matrix for C.

The equation 3.2 is called a *parity check equation*. Suppose $u = (u_1\, u_2\, ...\, u_k)$ is a message k-tuple which we wish to embed into a codeword x of length n. If the information symbols occur in the first k components of x, then $x = (u_1 u_2 ... u_k x_1 x_2 ... x_{n-k})$, where the last $n-k$ symbols are the "*check*" symbols. The check symbols can be uniquely determined by the information symbols since the codeword x satisfies the equation $Hx^T = 0$.

Example 3.1.3.**:** Consider the code over $\mathbb{F}_2$ with generator matrix

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

This code is systematic since the generator matrix is in standard form. The parity check matrix is derived from equation 3.3.

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

If the message space consists of 2-tuples, the code C may be obtained by substituting (00), (10), (01) and (11) for u in $x = uG$. The set of codewords is $C = \{0000, 1010, 0111, 1101\}$.

We can also obtain the set of codewords by using the parity check equations.

$$x_1 = u_1,$$

$$x_2 = u_2,$$

$$x_3 = u_1 + u_2,$$

$$x_4 = u_2.$$

The example above illustrates that the linear code can be completely described by giving a generator matrix or alternatively by giving a parity check matrix. These two descriptions are equivalent.

3.2 SYNDROME DECODING

We have viewed a code C as a subspace of a vector space $\mathbb{F}_q^n$. Thus, $\mathbb{F}_q^n$ is an abelian group under addition and C is a subgroup. We recall the concept of *cosets* from algebra.

Definition 3.2.1. *Let C be an $[n,k]$-linear code over $\mathbb{F}_q$. For any vector a, the set*

$$a + C = \{a + x : x \in C\}$$

is called a coset of C.

Every vector b is in some coset (for example, $b \in b + C$). Each coset contains q^k vectors.

Proposition 3.2.1. *For every $a, b \in \mathbb{F}_q^n$, a and b are in the same coset iff $(a-b) \in C$.*

Proof: Assume that $a - b \in C$ and denote $c = a - b$. Then, $a = c + b \in C + b$. Furthermore, $b \in C+b$. Thus, a and b are in the same coset. If a and b are in the same coset $C + x$ then $a = c + x$ and $b = c' + x$ for some $c, c' \in C$. Thus, $a - b = c - c' \in C$ as required. □

Proposition 3.2.2. *Two cosets are either disjoint or coincide.*

Proof: Assume the cosets $a + C$ and $b + C$ overlap. Then $x \in (a + C) \cap (b + C)$. Let $x = a + c = b + c'$, where $c, c' \in C$. Therefore $b = a + c - c' = a + c''$, $(c'' \in C)$. We conclude $b + C \subset a + C$. Similarly $a + C \subset b + C$. So $a + C = b + C$. □

The above proposition shows that the cosets of C partition the vector space $\mathbb{F}_q^n$. There are q^{n-k} different cosets.

$$\mathbb{F}_q^n = C \cup (a_1 + C) \cup (a_2 + C) \cup ... \cup (a_{q^{n-k}-1} + C) \tag{3.4}$$

Suppose the decoder receives the vector y. This vector must belong to one of the cosets, say $y = a_i + x$, $(x \in C)$. If the codeword x' was transmitted, the error vector is $e = y - x' = a_i + x - x' = a_i + x'' \in a_i + C$. We conclude that the possible error vectors are the vectors in the coset containing y. If $p < 1/2$ for the crossover probability, then, k bit errors are more likely than $k+1$ errors for each $k \in \{0, 1, \ldots, n\}$. Thus, the decoder should choose a minimum weight vector in the coset containing y as the error vector. If a coset has a unique minimum weight vector, the decoder has an easy choice. If there are several vectors of minimum weight in the coset $C + y$ the decoder selects one of these as the possible error. Such a minimum weight vector is called a *coset leader* and is selected for each coset prior to any communication taking place.

Definition 3.2.2. *The leader of a coset is defined to be a vector with the smallest Hamming weight in the coset, normally selected before any communication takes place.*

We assume that the a_i's in equation 3.4 are the coset leaders. We use a table called the "*standard array*" to describe the decoding process. The standard array of a code C is defined to be a table of vectors whose rows are the cosets of C. The first row is C itself with the zero vector in the first column. The first entry of any other row contains a coset leader and the remainder of the row is constructed by adding this leader to the codewords in the first row to obtain the other vectors in the coset.

Example 3.2.1. Consider the code over $\mathbb{F}_2$ with generator matrix

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

This code is systematic. The parity check matrix is

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

The first row of the standard array consists of the codewords. The minimum weight of the code is 2. We choose the vector (1000) of weight 1 to be the coset leader of the second coset and then add this vector to the codewords to fill out the remaining columns in the second row. We continue choosing vectors of weight 1 as coset leaders if they have not appeared in any previous coset. Since vectors (1000) and (0010) are in the first coset, we choose (0100) to be the coset leader of the second coset. A standard array for the code C generated by G is shown in the table below.

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

The decoder uses the standard array to decode the received vector y. It finds the position of y in the standard array. The decoder decides that the error vector e is the coset leader of the coset containing y. It decodes the vector y as a codeword $x = y - e$.

Decoding which uses a standard array is maximum likelihood decoding. The decoder can locate the coset containing y by computing the vector

$$S = Hy^T \tag{3.5}$$

where T denotes transpose. The vector S is called the *syndrome* of y.

Definition 3.2.3. *Let C be an $[n,k]$- linear code over $\mathbb{F}_q^n$ and let H be a parity-check matrix for C. Then for every $y \in \mathbb{F}_q^n$ the syndrome of y is defined to be the word*

$$S = Hy^T \in \mathbb{F}_q^{n-k}.$$

So S is a column vector of length $n-k$. The syndrome of y is zero if and only if y is a codeword by the definition of the code. Suppose the codeword x is transmitted and y is received. Then $y = x + e$ and

$$S = Hy^T = Hx^T + He^T = He^T. \tag{3.6}$$

Proposition 3.2.3. *Two vectors u and v are in the same coset of C if and only if they have the same syndrome.*

Proof: The vectors u and v are in the same coset iff $(u-v) \in C$ iff $H(u-v)^T = 0$ iff $Hu^T = Hv^T$. □

Corollary 3.2.1. *There is one-to-one correspondence between syndromes and cosets.*

Proof: The proof is immediate from the proposition 3.2.3..

The pure error detection scheme consists of testing if the syndrome is zero. When decoding uses the standard array, the decoder uses the following scheme:

Standard Array Decoding

Construct a standard array.

For each received vector y :

1. locate y in the standard array;

2. correct y to the codeword at the top of its column.

The standard array decoding scheme can be simplified by using the syndromes.

Syndrome Decoding

Set up a one-to-one correspondence between coset leaders and syndromes, and let H be the parity-check matrix.

For each received vector y :

1. compute the syndrome $S = Hy^T$;

2. locate the coset leader e associated with S;

3. correct y to $y - e$.

The decoding is correct if the true error is indeed a coset leader. If the true error is not a coset leader, then the decoder makes a decoding error and outputs the wrong codeword.

3.3 THE BINARY GOLAY CODE

There are a total of four Golay codes. One of the codes is still in use on the Voyager 1 & 2 space missions launched in 1977 to transmit pictures from Jupiter and Saturn. Voyager 1 is presently at the outer edge of the solar system and is destined to plunge into deep space. The code used in the Voyager missions consists of 4096 codewords that are 24-bit long binary strings. It is capable of correcting all patterns of 3 or fewer bit errors per codeword. This code is a variant of one introduced in 1949 by electrical engineer Marcel J. E. Golay. *Golay's 1949* code is slightly different from the *Voyager code* in that the codewords are one bit shorter, but they can still tolerate errors up to three bits for each transmitted codewords. A special feature of Golay's code is that if four or more of bits in a codeword get changed then we are guaranteed to incorrectly identify the codeword. Codes with this property are called

perfect, and are rather rare. Golay's code is known as the *perfect binary Golay code* and the Voyager code is called the *extended binary Golay code*.

Definition 3.3.1. *The extended Golay code C_{24} is a linear $[24, 12, 8]$ binary code with generator matrix :*

$$G = \left[\begin{array}{c|cccccccccccc|cccccccccccc} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{array}\right]$$

.

The extended Golay code C has minimum distance 8 and has a code rate of exactly $R = \frac{1}{2}$. The weight of every codeword is a multiple of 4, and C is invariant under a permutation of coordinates that interchanges the two halves of each codeword. The distribution of codewords by weight is given in the following table:

0	8	12	16	24
1	759	2576	759	1

Lemma 3.3.1. *C_{24} is self-dual. $C_{24} = C_{24}^{\perp}$.*

Since C_{24} is self-dual, the parity-check matrix H is equal to the generator matrix G. Automorphisms of codes are often used to construct efficient decoding schemes for

the given code. Although we will not use automorphisms in the decoding algorithms of our implementation we would like to mention the remarkable property of the extended Golay code that its full automorphism group is the five-transitive Mathieu group M_{24}, well-known to group theorists as one of the 26 sporadic simple groups (See [5]).

3.4 REED-MULLER CODES

Reed-Muller codes are among the oldest and best known families of codes. They were discovered by D. E. Muller in 1954 and the first decoding algorithm was devised by I. S. Reed, also in 1954. A Reed-Muller code was used by *Mariner 9* to transmit black and white photographs of Mars in 1972. One of the major advantages of Reed-Muller codes is their relative simplicity to encode messages and decode received transmissions. Reed-Muller codes are simple in construction and rich in structural properties.

Reed-Muller codes can be defined in terms of *Boolean functions*. We will define codes of length $n = 2^m$.

Definition 3.4.1. *A Boolean function is a function of the form* $f : \mathbb{F}^m \to \mathbb{F}$*, where* $\mathbb{F} = \mathbb{F}_2$ *is the finite field of order 2, and* $\mathbb{F}^m = \mathbb{F} \oplus \mathbb{F} \oplus \cdots \oplus \mathbb{F}$ *is a vector space of dimension* m *over* $\mathbb{F}$*.*

Of course, $\mathbb{F} = \mathbb{F}_2$ is endowed with the two binary operation "+" and "$\cdot$" which are integer addition and multiplication modulo 2. $V = \mathbb{F}^m$ is the set of all binary m-tuples. Let $v = (v_1, v_2, ..., v_m) \in \mathbb{F}^m$. Any function $f(v) = f(v_1, v_2, ..., v_m)$ which takes on the values 0 or 1 is a Boolean function. Such a function can be represented by a table, which gives the value of f at all possible 2^m instances of the arguments.

Example 3.4.1. The following table defines one such Boolean function f when $m = 3$.

v_1	v_2	v_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	0

The first three values on each row represent the argument of f and the fourth entry of each row represents the value of f at that argument.

The tabular form of a Boolean function is often called a *truth table* because of the connection between Boolean functions and logic. The last column of the truth table is a binary vector of length $n = 2^m$ which is often denoted by f. The last column can be filled in an arbitrarily way using 0's and 1's, so there are 2^{2^m} Boolean functions of m variables. The Boolean domain most often seen in the field of logic is $\mathbb{B} = \{T, F\}$, where T stands for true and F stands for false. The correspondence $\mathbb{B} = \{T, F\} \longleftrightarrow \{1, 0\} = \mathbb{F}_2$ is clear.

Example 3.4.2. There are $2^4 = 16$ two-variable Boolean functions $f(p, q)$. In this example, we define three commonly-used, two-variable Boolean functions.

The first function we define is *"p and q"*. For the moment, call this function $a(p, q)$. We have: $a(p, q) = 0$ unless both $p = 1$ and $q = 1$, in which case $a(p, q) = 1$. The function $a(p, q)$ is denoted by $p \wedge q$. We write $0 \wedge 0 = 0$, $0 \wedge 1 = 0$, $1 \wedge 0 = 0$, and $1 \wedge 1 = 1$.

The next function is *"p or q"*. Again, for the moment, call this function $b(p, q)$. We have: $b(p, q) = 1$ unless both $p = 0$ and $q = 0$, in which case $b(p, q) = 0$. The

function $b(p,q)$ is denoted by $p \vee q$. We write $0 \vee 0 = 0$, $0 \vee 1 = 1$, $1 \vee 0 = 1$, and $1 \vee 1 = 1$.

The last example is the *exclusive or* function. It is denoted by $p \oplus q$. Here, $p \oplus q = 0$ if $p = q$ and $p \oplus q = 1$ if $p \neq q$.

p	q	$p \wedge q$
0	0	0
0	1	0
1	0	0
1	1	1

p	q	$p \vee q$
0	0	0
0	1	1
1	0	1
1	1	1

p	q	$p \oplus q$
0	0	0
0	1	1
1	0	1
1	1	0

If the usual logical operations "$\wedge$", "$\vee$", "$\neg$ (negation)", and "$\oplus$" are written in terms of the operations "$+$" and "$\cdot$" of the binary field $\mathbb{F}_2$, then we have:

1. f **and** g $\quad f \wedge g = f \cdot g = fg$,
2. f **or** g $\quad f \vee g = f + g + fg$,
3. **not** f $\quad \neg f = \overline{f} = 1 + f$,
4. f **exclusive or** g $\quad f \oplus g = f + g$.

A Boolean function can be written down from its truth table in terms of "$\wedge$", "$\vee$", and "$\neg$" (negation) by means of the well known *disjunctive normal form* [1] . In Example 3.4.1., $f = v_3 v_2 \overline{v_1} \vee \overline{v_3}\ \overline{v_2}\, v_1$ corresponding to the rows of the truth table where f is 1. This is called the *disjunctive normal form* for f. Using the equivalences above in terms of the field $\mathbb{F}_2$ operations, this simplifies to $f = v_1 + v_1 v_2 + v_1 v_3 + v_2 v_3$. It

is clear that in this way any Boolean function can be expressed as a linear combination of the 2^m monomials/vectors:

$$1, v_1, v_2, \ldots, v_m \ ; \ v_1v_2, v_1v_3, \ldots, v_{m-1}v_m \ ; \ \ldots ; \ v_1v_2\cdots v_m$$

with coefficients which are 0 or 1. Therefore, any Boolean function is also a polynomial in $v_1, v_2, ..., v_m$. The 2^m monomials displayed above are linearly independent over $\mathbb{F}_2$ and span the space of all 2^{2^m} Boolean functions. These 2^m monomials/vectors form a basis for the Reed-Muller code.

Let $v = (v_1, v_2, ..., v_m)$ denote a vector in $\mathbb{F}^m$ and let f be the vector of length 2^m obtained from a Boolean function $f(v_1, v_2, ..., v_m)$.

Definition 3.4.2. *The r^{th} order binary Reed-Muller code $\mathcal{R}(r, m)$ of length $n = 2^m$, for $0 \leq r \leq m$, is the set of all vectors f, where $f(v_1, v_2, ..., v_m)$ is a Boolean function which is a polynomial of degree at most r in $v_1, \ldots, v_m$ over $\mathbb{F}_2$.*

Example 3.4.3. The **first order** Reed-Muller code of length 8 contains 16 codewords which are the collection of all binary linear combinations of $1, v_1, v_2, v_3$. The codewords are shown in the following table:

0	00000000
v_1	00001111
v_2	00110011
v_3	01010101
$v_1 + v_2$	00111100
$v_1 + v_3$	01011010
$v_2 + v_3$	01100110
$v_1 + v_2 + v_3$	01101001
1	11111111
$1 + v_1$	11110000
$1 + v_2$	11001100
$1 + v_3$	10101010
$1 + v_1 + v_2$	11000011
$1 + v_1 + v_3$	10100101
$1 + v_2 + v_3$	10011001
$1 + v_1 + v_2 + v_3$	10010110

All the codewords of $\mathcal{R}(1, m)$ except for 0 and 1 have weight 2^{m-1}.

In general the r^{th} order Reed-Muller code consists of all linear combinations of the vectors corresponding to the monomials :

$$1, v_1, \ldots, v_m,\ v_1v_2, v_1v_3, \ldots, v_{m-1}v_m,\ \ldots\ v_1v_2\cdots v_r, \ldots, v_{m-r+1}\cdots v_{m-1}v_m$$

(all monomials of degree up to r), which therefore form a basis for the code. There are

$$k = 1 + \binom{m}{1} + \binom{m}{2} + ... + \binom{m}{r}$$

such basis vectors for $\mathcal{R}(r, m)$. So k is the dimension of the code.

Example 3.4.4. The following table contains 8 possible basis vectors for Reed-Muller code of length 8 when $m = 3$:

1	11111111
v_1	00001111
v_2	00110011
v_3	01010101
v_1v_2	00000011
v_1v_3	00000101
v_2v_3	00010001
$v_1v_2v_3$	00000001

The basis vectors for the r^{th} order Reed-Muller code of length 8 are shown in the table below:

Order r	Rows
0	1
1	1-4
2	1-7
3	1-8

Theorem 3.4.1. *(See [1])* $\mathcal{R}(r, m)$ *has minimum distance* 2^{m-r}.

Chapter 4

Implementation

In this project we perform data transmission across noisy channels and recover the message first by using the [24,12,8] Golay code, and then by using the [32,6,16] first-order Reed-Muller code $\mathcal{R}(1,5)$. It is known that different codes are optimal for different applications. Our goal is to determine which code among the above two is more efficient for text message transmission, by applying the two codes to exactly the same data with the same channel error bit probabilities p. By the efficiency of a code we mean the combination of reliability and speed of a code for a particular application. We compare the error-correcting capability and the practical speed of the Golay code and the first-order Reed-Muller code. We employed "A Programming Language" (APL) that was first described in a 1962 book of the same name by Kenneth E. Iverson. APL programs tend to be very short in comparison with implementations using other languages.

4.1 IMPLEMENTING $\mathbf{C_{24}}$

The extended Golay code C_{24} is a linear $[24, 12, 8]$-code with generator matrix G, which is the same matrix as the one in definition 3.3.1 except that the first column of the earlier matrix is removed and placed as the last column. This was done so that the generator matrix is systematic, i.e. with a 12×12 identity matrix appearing on

the leftmost part of G to facilitate decoding. The submatrix of A of G corresponding to rows 1-11 and columns 13-23 is a circulant matrix with first row 1 0 1 1 1 0 0 0 1 0 , and was constructed by using program **rr7** in Table 1. The single line instruction in Table 2. constructs the 12×12 identity matrix.

Table 1 (Matrix A)	
[0]	**rr7**
[1]	$A \leftarrow 11\ 11\ \rho\, 0\ \diamond\ i \leftarrow 1$
[2]	$x \leftarrow 10111000101$
[3]	$a1 : A[i;] \leftarrow (-i)\phi x$
[4]	$\rightarrow a1 \times \iota\, 11 \geq i \leftarrow i + 1$

Table 2 (Matrix I_{12})	
[1]	$id \leftarrow (i12) \circ . = i12$

The systematic generator matrix G we used for the [24,12,8] extended Golay code C_{24} is :

$$G = \left[\begin{array}{cccccccccccc|cccccccccccc} 1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&1&0&1\\ 0&1&0&0&0&0&0&0&0&0&0&0&0&1&1&0&1&1&1&0&0&0&1&1\\ 0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&0&0&1\\ 0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&0&1\\ 0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&0&1\\ 0&0&0&0&0&1&0&0&0&0&0&0&0&0&0&1&0&1&1&0&1&1&1&1\\ 0&0&0&0&0&0&1&0&0&0&0&0&1&0&0&0&1&0&1&1&0&1&1&1\\ 0&0&0&0&0&0&0&1&0&0&0&0&1&1&0&0&0&1&0&1&1&0&1&1\\ 0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&0&1&0&1&1&0&1\\ 0&0&0&0&0&0&0&0&0&1&0&0&0&1&1&1&0&0&0&1&0&1&1&1\\ 0&0&0&0&0&0&0&0&0&0&1&0&1&0&1&1&1&0&0&0&1&0&1&1\\ \hline 0&0&0&0&0&0&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&0 \end{array}\right].$$

The 12×24 matrix G generates the code, and since C_{24} is self-dual, the parity check matrix H is identical with G. The length of each codeword is 24 bits. The rank

of the generator matrix is 12. There are $2^{12} = 4096$ codewords in the code. We use program **rr6** in table 3 to construct the Golay code:

Table 3 (The Golay Code)	
[0]	**rr6**
[1]	$C \leftarrow 4096\ 24\ \rho\ 0\ \diamond\ i \leftarrow 1$
[2]	$a1 : x \leftarrow (12\rho 2)\top i$
[3]	$v \leftarrow 2\vert x + . \times G$
[4]	$c[i + 1;] \leftarrow v$
[5]	$\rightarrow a1 \times \iota\ 4095 \geq i \leftarrow i + 1$

4.1.1 Coset leaders for Golay C_{24}

Since the minimum weight of $C = C_{24}$ is 8, if $w \leq 3$, no coset of C in $V = \mathbb{F}_2^{24}$ contains more than one vector of weight w. Thus for each vector v of weight $w \leq 3$ v is a coset leader. This accounts for

$$1 + \binom{24}{1} + \binom{24}{2} + \binom{24}{3} = 2325$$

cosets and coset leaders. For each vector $v \in V$ of weight ≤ 3 we compute the syndrome

$$s = Hv^T \in F_2^{12}$$

and store v as a coset leader in the i^{th} row of matrix *cosl*, where i is the base 2 representation of s. This way during decoding, once we have computed a syndrome s we can immediately locate the corresponding coset leader by looking at the corresponding row of table *cosl*. Further, for each of the remaining $1771 = 2^{12} - 2325$ cosets we observe that there are exactly six vectors of weight 4 which are members of the coset.

We select the lexicographically smallest vector v among these, compute the syndrome s and store v in row i of *cosl* where again i is the base 2 representation of s. Program **rr9** (Table 6) computes the syndromes and places them in table *cosl* as discussed.

Table 6 (rr9)	
[0]	**rr9**
[1]	$m \leftarrow 2 \star 12 \diamond n \leftarrow 24 \diamond k \leftarrow 1$
[2]	$cosl \leftarrow (m, 24)\,\rho\, 9$
[3]	$cosl[1;] \leftarrow 24\,\rho\, 0$
[4]	$j \leftarrow 2$
[5]	$a1 : nk \leftarrow n, k$
[6]	$r \leftarrow nk[2]\,!\,nk[1]$
[7]	$i \leftarrow 1$
[8]	$a2 : set \leftarrow nk\ duper\ i$
[9]	$v \leftarrow 24\,\rho\, 0 \diamond v[set] \leftarrow 1$
[10]	$s \leftarrow 2 \vert H + . \times v$
[11]	$j \leftarrow 1 + 2 \perp s$
[12]	$cosl[j;] \leftarrow v$
[13]	$\rightarrow a2 \times \iota\, r \geq i \leftarrow i + 1$
[14]	$\rightarrow a1 \times \iota\, 4 \geq k \leftarrow k + 1$

4.1.2 Processing a basic Golay message block

Since the goal was to compare the two codes C_{24} and $\mathcal{R}(1,5)$ of lengths 24 and 32 with corresponding message lengths 12 and 6 respectively, we decided to use a *basic message block* of $96 = \text{lcm}\{24, 32\}$ bits. The 96 bits accommodate exactly 12

8-bit "bytes" and each byte encodes a single alphabetic character from an enhanced alphabet of 256 possible characters. The full text to be tested consisted of the latex source file of this thesis of approximately 65,000 characters.

We divide a text message into blocks of 12 characters if the number of characters is divisible by 12. If the number of characters is not divisible by 12, we add some random characters so the text length is divisible by 12. We convert letters into numbers and numbers into sequences of bits of length 8. Each block of 12 alphabetic characters is transformed into a string of 96 bits by program **code1**, which for the Golay code is reshaped into an 8×12 binary matrix $\mathbf{x}$ (Table 4). The 12 text characters converted to an 8×12 binary matrix will be interpreted as 8 messages to be encoded as 8 codewords in the Golay code.

Table 4 (code1)	
[0]	$\mathbf{x} \leftarrow$ **code1** z
[1]	$\mathbf{x} \leftarrow 8\ 12\ \rho,\ \phi\ (8\rho 2)\top\Box av\iota z$

We encode the original message block by multiplying matrix $\mathbf{x}$ by the generator matrix G. The resulting encoded message block is an 8×24 matrix $\mathbf{x}G$ which we label as **xg** (in the APL program(s).)

This encoded text message block **xg** is transmitted over the channel with bit error probability p. We use program **channel** (Table 5) to introduce bit errors. There are $8 \cdot 24 = 192$ bits in a codeword block. The number of errors introduced in a codeword block by the channel is $z = \lceil 192 \cdot p \rceil$. After transmitting the encoded text message block through the channel we obtain an 8×24 matrix which we label as **ye**. Matrix **ye** includes the initial Golay codewords possibly altered by z uniformly distributed random bit errors.

Table 5 (channel)	
[0]	$ye \leftarrow xg$ **channel** $p; n; v; t$
[1]	$n \leftarrow \lceil 192 \times p$
[2]	$v \leftarrow, xg$
[3]	$t \leftarrow n?192$
[4]	$v[t] \leftarrow\sim v[t]$
[5]	$ye \leftarrow 8\ 24\rho v$

We use Syndrome Decoding to decode the message transmitted through the channel. As we saw earlier, for this code, it is practical and relatively easy to compute and store the 4096 coset leaders.

Since the Golay code is self-dual, the parity-check matrix H is equal to the generator matrix G. We compute the 8 syndromes of "received" matrix **ye** as the 8 columns of the product

$$H\mathbf{ye}^T$$

We locate the errors by using the one-to-one correspondence of coset leaders and syndromes (Using the look-up table *cosl* stored in the memory of the computer). An 8×24 error matrix **err** is formed which contains 8 coset leaders corresponding to the 8 syndromes we computed. After adding matrix **ye** to matrix **err** we obtain an 8×24 matrix **y** with errors corrected. We perform the final steps of the decoding process by i) obtaining the 8 Golay message vectors $\mathbf{x}'$ by retrieving the first 12 columns of matrix **y** and ii) using the program **decode1** (Table 8). Program decode1 converts sequences of bits into numbers and numbers into letters.

Program **run** (Table 7) is responsible for encoding a text message of 12 characters, transmitting this encoded text message through the channel, correcting the errors

introduced by the channel and converting back into a text message. Program *run* is incorporated into program **longrun** (Table 9) which works on texts containing more than 12 characters.

Table 7 (run)	
[0]	$w \leftarrow z$ **run** p
[1]	$x \leftarrow code1\ z$
[2]	$xg \leftarrow\ 2\vert x + . \times G$
[3]	$ye \leftarrow xg\ channel\ p$
[4]	$s \leftarrow 2\vert H + . \times \phi\, ye$
[5]	$t \leftarrow 1 + 2\perp s$
[6]	$err \leftarrow cosl[t;]$
[7]	$y \leftarrow 2\vert err + ye$
[8]	$xp \leftarrow y[;\iota 12]$
[9]	$w \leftarrow decode1\ xp$

Table 8 (decode1)	
[0]	**decode1** $v; t$
[1]	$t \leftarrow 2\perp\phi 12\ 8\rho\, , v$
[2]	$z \leftarrow \Box av[t]$

Table 9 (longrun)	
[0]	$textout \leftarrow text$ **longrun** $p; xxx;$ $n; m; mx; my$
[1]	$t1 \leftarrow \Box ts$
[2]	$xxx \leftarrow text \diamond textout \leftarrow 0\rho'\ \ '$
[3]	$n \leftarrow\ \rho\, xxx$
[4]	$m \leftarrow 12\vert n$
[5]	$\rightarrow a1 \times \iota m = 0$
[6]	$xxx \leftarrow xxx, (12 - m)\rho\Box av[167]$
[7]	$a1 : mx \leftarrow 12 \uparrow xxx$
[8]	$my \leftarrow\ mx\ run\ p$
[9]	$textout \leftarrow textout, my$
[10]	$xxx \leftarrow 12 \downarrow xxx$
[11]	$\rightarrow a1 \times \iota 0 \leq \rho xxx$
[12]	$t2 \leftarrow \Box ts$
[13]	$t1$
[14]	$t2$

4.2 IMPLEMENTING $\mathcal{R}(1, 5)$ CODE R

The Reed-Muller code $\mathcal{R}(1, 5)$ has been chosen for the comparison of the error-correcting capability and speed of the text message transmission with the Golay

code. $\mathcal{R}(1,5)$ contains vectors corresponding to all polynomials in the binary variables $1, v_1, v_2, ..., v_5$ of degree less or equal to 1. The length of the code is $n = 2^5 = 32$, the dimension is $k = 1 + \binom{5}{1} = 6$, and the minimum distance is $d = 2^{5-1} = 2^4 = 16$. The Reed-Muller code $\mathcal{R}(1,5)$ consists of 64 codewords of the form

$$a_0 1 + a_1 v_1 + a_2 v_2 + a_3 v_3 + a_4 v_4 + a_5 v_5,$$

where $a_i \in \mathbb{F}_2$, for $i = 0, 1, ..., 5$.

The generator matrix G_1 of the Reed-Muller code $\mathcal{R}(1,5)$ is the 6×32 matrix whose rows are the vectors corresponding to $1, v_1, v_2, ..., v_5$. We transform G_1 into the generator matrix G in standard form by elementary row operations and column permutations, to simplify the decoding process. G generates an equivalent code $\mathbf{R}$.

$$G = \begin{bmatrix} 10000011001111001011110011000011 \\ 01000000000000000111111111111111 \\ 00100000011111110000000011111111 \\ 00010001111010011110100101101001 \\ 00001011101100110011001100110011 \\ 00000110111001101110011001100110 \end{bmatrix}$$

.

4.2.1 Processing a basic message block in R

The front- and back-end processing of a basic 12 alphabetic character block is almost identical to what was done in the Golay code case, except that the basic 96 bit block is here reshaped to a 16×6 matrix $\mathbf{x}$ of 16 RM messages. Encoding is straight forward and is achieved by multiplying $\mathbf{x}$ by G :

$$\mathbf{y} = \mathbf{x}G$$

resulting in a 16×32 matrix $\mathbf{y}$ of 16 RM codewords. Program **channel** works exactly like in the Golay case to infuse bit errors in $\mathbf{y}$ with a given crossover probability p. After transmitting the encoded message through the channel we obtain the 16×32 matrix **ye**. This matrix contains 16 vectors of length 32 each, possibly altered by channel errors.

Here, processing **ye** diverges significantly from what we did in the Golay case. Note that here the dimension of the code is 6, and the length 32. That means that $|\mathbf{R}| = 64$, and that there are in all $2^{32-6} = 2^{26} = 67,108,864$ cosets. This is a huge space requirement for storing a syndrome to coset leader table. Even if this were feasible, it is certainly impractical, and we abandon the idea of decoding by computing syndromes. At this point we adopt two different methods for decoding which we will discuss briefly below:

4.2.2 RM decoding method α

For each row $\mathbf{u}$ of **ye** we compute the coset $\mathbf{u} + \mathbf{R}$, and choose the (or a) lowest weight vector in the coset as the error vector **e**. If 7 or fewer bit errors have occurred for this $\mathbf{u}$ there will be a unique vector of lowest weight in the coset, since C corrects up to 7 errors. If more than 7 errors have occurred in $\mathbf{u}$ there will be more than one lowest weight vectors in the coset, and there is no guarantee that the one we select is indeed the actual error vector. In the latter case we may commit a decoder error. We recover the RM codeword corresponding to $\mathbf{u}$ as $\mathbf{u} + \mathbf{e}$.

We perform this process for all of the 16 row vectors of **ye**. This is of course a high time complexity task resulting in a much slower process than the corresponding Golay step.

We use program **rmdecode** to correct errors introduced by the channel. Program **rmdec** is incorporated into program **rmdecode**. The latter constructs the coset of

each vector in **ye** and locates a/the vector of least weight (coset leader). The errors get corrected by adding to each row vector in **ye** its coset leader. After correcting the errors program **rmdecode** retrieves the first 6 columns of 16×32 corrected matrix. The resulting 16×6 RM message matrix is appropriately reshaped into an 8×12 bit matrix. We then use program **decode1** to convert sequences of bits into numbers and numbers into letters.

Program **run** is responsible for encoding a text message of 12 characters, transmitting this encoded text message through the channel, correcting the errors introduced by the channel and converting back into a text message. Program **run** is incorporated into program **longrun** which works on texts containing more than 12 characters.

4.2.3 RM decoding method β

In this decoding method we use the special properties of first order RM codes in connection to the Hadamard transform. For proofs of the statements we rely on [1][pp 413-420]. The process is also known as the Discrete Fourier transform and can be made much faster by employing the Fast Fourier transform (FFT), which we do not implement here.

Consider the mapping $F : \{0,1\} \to \{1,-1\}$ defined by $F(x) = (-1)^x$, thus $F(0) = 1$ and $F(1) = -1$. Extend the domain of F to binary vectors and matrices by applying F elementwise.

Proposition 4.2.1. *Let S be any 5×32 submatrix of G, and U the code spanned by S over $\mathbb{F}_2$. If M_U is the matrix whose rows are the 32 vectors of U, then $H = F(M_U)$ is a Hadamard matrix, i.e.*

$$HH^T = 32 \cdot I_{32}$$

In particular, we select the top 5 rows of G for S, and develop M_U by placing $v \cdot G$ in the i^{th} row of M_U where v is the vector representing i in its base 2 representation. We then compute $H = F(M_U)$.

Proposition 4.2.2. *Suppose that $v \in F_2^5$, and that i is the integer whose base 2 representation is v. Let $u = v \cdot S \in U$ and let $f = F(u)$. Then $f \cdot H$ is a 1×32 vector with 32 in the i^{th} position and zeroes everywhere else.*

Proof: $u = v \cdot S$ is the i^{th} row of M_U, so that f is the i^{th} row of H. Now, all rows of H are orthogonal to f except for f itself, and $(f, f) = 32$. □

We now present the following facts which along with Proposition 4.2.2 allow us to decode received vectors in $\mathcal{R}(1, 5)$.

Remark 4.2.1. *We make three observations :*

i) Let v_6 be the sixth row of G, and u as above. Then $F(u + v_6)$ has -32 in position i,

ii) If $x \in U$, and $y = x + e$, where $wt(e) \leq 7$, then the maximum entry of $F(y) \cdot H$ will occur in position i and will be positive, with all other entries strictly less in absolute value than this maximum.

iii) If $y = v_6 + u + e$, $u \in U$, $wt(e) \leq 7$, then the maximum in absolute value of $F(y)$ will occur in position i, and will be negative, with the other entries strictly less in absolute value than this maximum.

Table 10 (rmencode)	
[0]	$yy \leftarrow$ **rmencode** $xx; tt$
[1]	$tt \leftarrow 16\ 6\rho 0$
[2]	$tt[\iota 8;] \leftarrow xx[;\iota 6]$
[3]	$tt[8 + \iota 8;] \leftarrow xx[; 6 + \iota 6]$
[4]	$yy \leftarrow 2 \vert tt + . \times GRM$

Table 11 (channel)	
[0]	$ye \leftarrow yy$ **channel** $p; m; sh; n; v; t$
[1]	$m \leftarrow x/sh \leftarrow \rho yy \diamond n \leftarrow \lceil m \times p$
[2]	$v \leftarrow, yy$
[3]	$t \leftarrow n?m$
[4]	$v[t] \leftarrow\sim v[t]$
[5]	$ye \leftarrow sh\rho v$

Table 12 (rmdecode)	
[0]	$tt \leftarrow$ **rmdecode** $ye; i; zz$
[1]	$zz \leftarrow 16\ 32\rho 0 \diamond \iota \leftarrow 1$
[2]	$a1 : zz[\iota;] \leftarrow rmdec\ ye[\iota;] \diamond \rightarrow a1 \times \iota 16 \geq \iota \leftarrow \iota + 1$
[3]	$zz \leftarrow zz[;\iota 6] \diamond\ tt \leftarrow 8\ 12\rho 0$
[4]	$tt[;\iota 6] \leftarrow zz[\iota 8;] \diamond\ tt[; 6 + \iota 6] \leftarrow zz[8 + \iota 8;]$

Table 13 (rmdec)	
[0]	$z \leftarrow$ **rmdec** $y; cosy; pi; t$
[1]	$cosy \leftarrow 2 \vert (64\ 32\rho y) + C$
[2]	$pi \leftarrow\uparrow t \leftarrow +/cosy$
[3]	$z \leftarrow 2 \vert y + cosy[pi[1];]$

Table 14 (runr)	
[0]	$zz \leftarrow mx$ **runr** $p; yy; ye$
[1]	$yy \leftarrow rmencode\ code1\ mx$
[2]	$ye \leftarrow yychannelp$
[3]	$zz \leftarrow decode1\ rmdecode\ ye$

Table 15 (longrunr)	
[0]	$textout \leftarrow text$ **longrunr** $p; xxx; n; m; mx; my$
[1]	$t1 \leftarrow \Box ts$
[2]	$xxx \leftarrow text\ \diamond\ textout \leftarrow 0\ \rho'\ '$
[3]	$n \leftarrow\ \rho\ xxx$
[4]	$m \leftarrow 12 \vert n$
[5]	$\rightarrow a1 \times \iota m = 0$
[6]	$xxx \leftarrow xxx, (12 - m)\rho \Box av[167]$
[7]	$a1 : mx \leftarrow 12 \uparrow xxx$
[8]	$my \leftarrow\ mx\ runr\ p$
[9]	$textout \leftarrow textout, my$
[10]	$xxx \leftarrow 12 \downarrow xxx$
[11]	$\rightarrow a1 \times \iota\ 0 \leq \rho xxx$
[12]	$t2 \leftarrow \Box ts$
[13]	$t1$
[14]	$t2$

Chapter 5

Conclusions

We perform transmission of the text containing 65000 characters across a noisy channel and recover the message first by using the [24,12,8] Golay code, and then using the [32,6,16] first-order Reed-Muller code. We have the ability to assign any error bit probability to the channel. The main objective of this thesis is to determine which code among the above two is more efficient for text message transmission, first in terms of error correcting capability, and secondly in terms of speed. We apply the two codes to the same text with the same channel error bit probabilities gradually increasing p until 50% is reached. At channel error probability of 5% both the Golay and the the Reed-Muller codes performed error correction at 100%. Noticeably, at 6% error bit probability the Golay code fails to meet the 100% correction of errors introduced by the channel. The Reed-Muller code continues to maintain 100% correction until the channel has introduced the 15% bit errors. The graph shows that at 10% error bit probability 20% of text character errors will not be corrected by the Golay code. The error bit probability can be doubled for the Reed-Muller code to meet the same outcome of error correction as the Golay code. We can see from the graph that at a 20% error-bit probability 80% of text character errors will not be corrected by the Golay code. It will require us to increase the number of bit errors by the additional 10% for the Reed-Muller code to fail to correct 80% of the text character errors

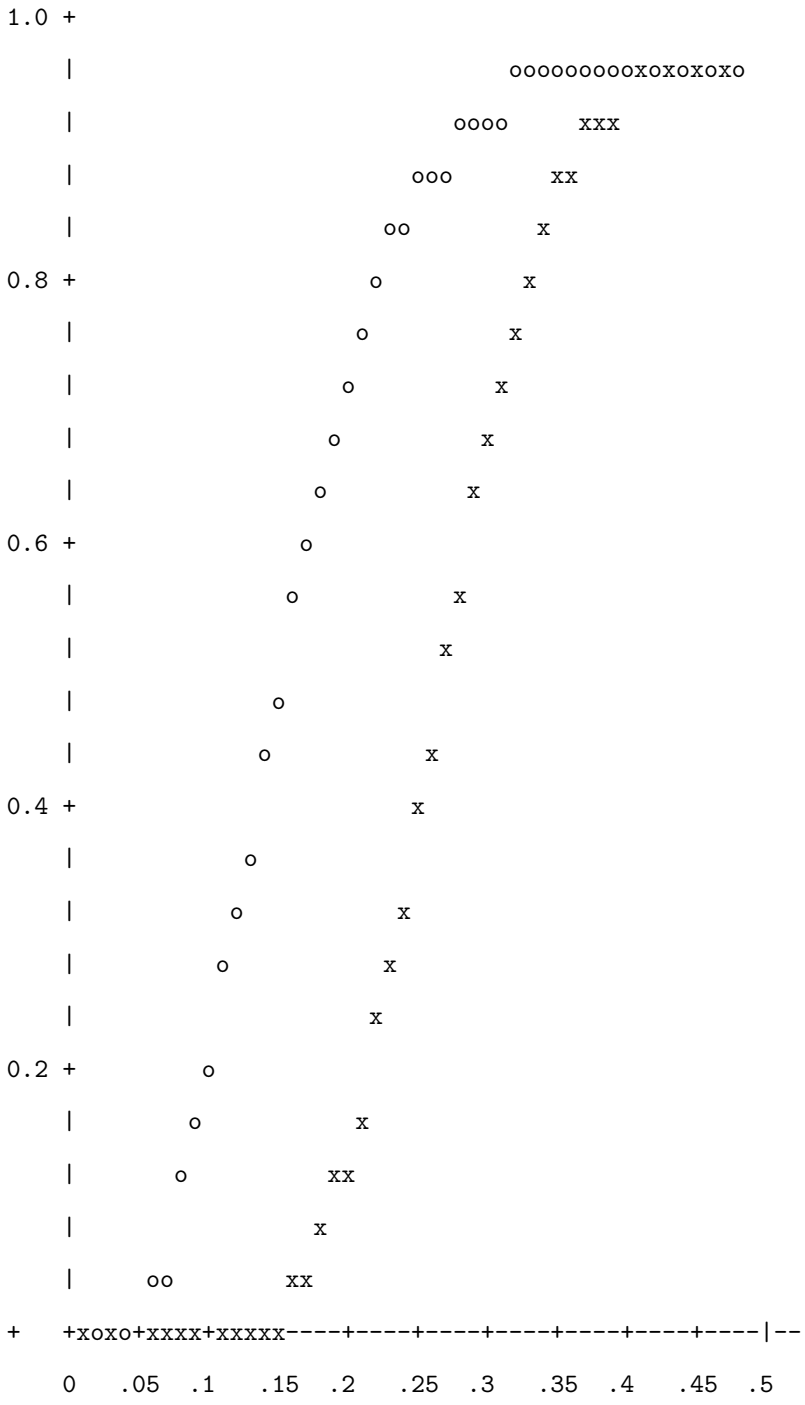


Figure 5.1: Comparing error correcting capabilities: o indicates the Golay code, x indicates the Reed-Muller code

introduced by the channel. The results obtained are consistent with our expectations since the Reed-Muller code is capable of correcting up to 7 errors while the Golay code can only correct up to 3 errors per codeword. The process of encoding the text containing 65000 characters, translating over the channel with error bit probability 0.01 and decoding back to the text message by using the Golay code took an average of 1.856 seconds. Then we repeated the cycle by using the Reed-Muller code on the same text message with the same error bit probability. The whole process was done in an average of 10.056 seconds. Using the same text with $\mathcal{R}(1,5)$ and decoding method β ran at an average of 8.125 seconds, an improvement of about 20% in speed over method α. The RM speed could be increased further if one used Fast Fourier Transform, but we did not explore FFT in this thesis. We can see that the practical speed of the Golay code is definitely better in comparison with the speed of the Reed-Muller code due to the difference in the decoding methods. We use the lookup table stored in the memory of the computer for the decoding purposes of the Golay code. With the Reed-Muller code we have to construct the entire coset containing all its vectors every time we need to correct an error. The Reed-Muller code is definitely time consuming, but is much preferable in a situation where the channel is very noisy. As a result we can see there is is a definite tradeoff between speed and error correcting capability.

Chapter 6

APPENDIX

[0]	$set \leftarrow nk$ **duper** $x; s; m; q; t; i; eps$
[1]	$eps \leftarrow 10 \star -12 \diamond set \leftarrow \iota 0 \diamond m \leftarrow nk\,[1] - i \leftarrow 1 \diamond q \leftarrow nk[2] - 1$
[2]	$a1 : t \leftarrow m\ choose\ q$
[3]	$\rightarrow a2 \times \iota\, t < x$
[4]	$m \leftarrow m - 1$
[5]	$set \leftarrow set, i$
[6]	$q \leftarrow q - 1$
[7]	$\rightarrow 0 \times \iota q < 0$
[8]	$i \leftarrow i + 1$
[9]	$\rightarrow a1$
[10]	$a2 : x \leftarrow \lfloor eps + x - t$
[11]	$i \leftarrow i + 1$
[12]	$m \leftarrow m - 1$
[13]	$\rightarrow a1$

BIBLIOGRAPHY

[1] F.J. MacWilliams and N.J.A. Sloane, The theory of Error-Correcting Codes, North-Holland Math. Lib., Elsevier Sc. B.V. , Amsterdam, 1977, isbn 0444851933, pp. i – 762.

[2] S.A. Vanstone and P.C. van Oorschot, An Introduction to Error Correcting Codes with Applications, Kluwer Academic Publishers, Boston, 1989, isbn 0792390172, pp. i – 289.

[3] V. Pless, Introduction to the theory of Error-Correcting Codes, John Wiley and Sons, New-York, 1998, isbn 0471190470, pp. i – 207.

[4] J.H. van Lint, Introduction to Coding Theory, Springer-Verlag Berlin Heidelberg, 1982, isbn 3540641335, pp. i – 234.

[5] J.J. Rotman, An Introduction to the Theory of Groups, Springer-Verlag New-York, 1995, isbn 0387942858, pp. i – 513.