
MATH 291T

CODING THEORY

Spring 2009

Instructor : Stefaan Delcroix

Chapter 1

Introduction to Error-Correcting Codes

It happens quite often that a message becomes corrupt when the message is transmitted using some communication system (noise interference during cell phone calls, scratches on cd's, temporarily loss of signal in deep space programs). If we only transmit the message then it is impossible to recover the message if an error occurs during transmission.

For example, we have a channel that can transmit the symbols 0 and 1. We have two messages: yes=1 and no=0. We suppose that the channel is symmetric : the probability that an error occurs during transmission of a symbol, is a fixed number p (where $0 \leq p \leq 0.5$). When we transmit a message, the probability of receiving the original message is $1 - p$.

In order to recover the original message if an error occurs, we have to add some redundancy. We make our message longer : yes=11 and no=00. After transmission of a message, we receive the string $y_1y_2 \in \{00, 11, 01, 10\}$. In the table below, we list in the second (resp. third) column the probability $P(y_1y_2|00)$ (resp. $P(y_1y_2|11)$) that y_1y_2 is received when 00 (resp. 11) was transmitted while in the fourth (resp. fifth) column we list the probability $P(00|y_1y_2)$ (resp. $P(11|y_1y_2)$) that 00 (resp. 11) was sent if y_1y_2 is received.

y_1y_2	$P(y_1y_2 00)$	$P(y_1y_2 11)$	$P(00 y_1y_2)$	$P(11 y_1y_2)$
00	$(1 - p)^2$	p^2	$\frac{(1 - p)^2}{p^2 + (1 - p)^2}$	$\frac{p^2}{p^2 + (1 - p)^2}$
01 or 10	$p(1 - p)$	$p(1 - p)$	$\frac{1}{2}$	$\frac{1}{2}$
11	p^2	$(1 - p)^2$	$\frac{(1 - p)^2}{p^2 + (1 - p)^2}$	$\frac{p^2}{p^2 + (1 - p)^2}$

What does this imply? Say $p = 10\%$. If we receive the word 00 (resp. 11) then the probability that 00 (resp. 11) was transmitted is 98.78%. If we receive the word 01 or 10 then the probability

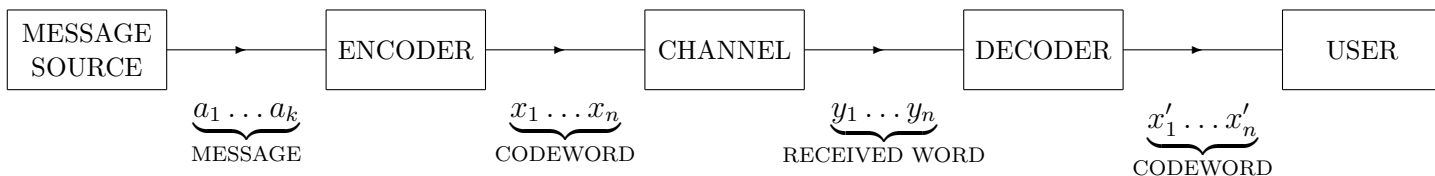
that 00 (resp. 11) was transmitted is 50%. Note that we are able to detect if exactly one error occurred but we are not able to correct the error.

So we add even more redundancy. Our messages are yes=111 and no=000. After transmission of a message, we receive the string $y_1y_2y_3 \in \{000, 111, 100, 010, 001, 011, 101, 110\}$. In the table below, we list in the second (resp. third) column the probability $P(y_1y_2y_3|000)$ (resp. $P(y_1y_2y_3|111)$) that $y_1y_2y_3$ is received when 000 (resp. 111) was transmitted while in the fourth (resp. fifth) column we list the probability $P(000|y_1y_2y_3)$ (resp. $P(111|y_1y_2y_3)$) that 000 (resp. 111) was sent if $y_1y_2y_3$ is received.

$y_1y_2y_3$	$P(y_1y_2y_3 000)$	$P(y_1y_2y_3 111)$	$P(000 y_1y_2y_3)$	$P(111 y_1y_2y_3)$
000	$(1-p)^3$	p^3	$\frac{(1-p)^3}{p^3 + (1-p)^3}$	$\frac{p^3}{p^3 + (1-p)^3}$
100, 010 or 001	$p(1-p)^2$	$p^2(1-p)$	$1-p$	p
011, 101 or 110	$p^2(1-p)$	$p(1-p)^2$	p	$1-p$
111	p^3	$(1-p)^3$	$\frac{p^3}{p^3 + (1-p)^3}$	$\frac{(1-p)^3}{p^3 + (1-p)^3}$

Say $p = 10\%$. If we receive the word 000 (resp. 111) then the probability that 000 (resp. 111) was transmitted is 99.86%. If we receive any of the words 100, 010 or 001 (resp. 011, 101 or 110) then the probability that 000 (resp. 111) was transmitted is 90% so we decode as 000 (resp. 111). Note that we are able to detect if exactly one error occurred and we are able to correct the error. Using this decoding algorithm, the probability of decoding correctly is 97.2%.

In general, transmitting information might undergo the following process.



We start with a message $a_1 \dots a_k$ containing k digits. We add some redundancy and end up with a codeword $x_1 \dots x_n$ containing n digits (so $n > k$). We transmit the codeword $x_1 \dots x_n$ but during transmission, errors may occur. So we receive a word $y_1 \dots y_n$. After analyzing this word, we decide that the original codeword was $x'_1 \dots x'_n$ (and hopefully this is indeed the original codeword). The user now retrieves the original message from the codeword $x'_1 \dots x'_n$.

We will concentrate mainly on the decoder : when we receive a word $y_1 \dots y_n$, how do change this word into a codeword (hopefully the codeword that was transmitted)? Clearly, we should choose the codeword that was most likely sent (so the codeword with the highest probability of being sent). This method is called *maximum likelihood decoding* and might be quite hard to implement. Instead, we find the codeword that is the closest to the received word (so we minimize the number of errors that occurred). This method is called *nearest neighbor decoding*.

Chapter 2

Basic Concepts and Properties

2.1 Definitions

Definition 2.1

- (1) An *alphabet* is a set of symbols or characters with at least two elements. In this course, alphabets will always be finite fields.
- (2) Let A be an alphabet and $n \in \mathbb{N}$.
 - (a) A^n is the set of all n -tuples of elements of A . So

$$A^n = \{(a_1, \dots, a_n) : a_1, \dots, a_n \in A\}$$

We call A^n the *codespace* and refer to elements of A^n as *words* or *vectors*.

If $\mathbf{a} = (a_1, \dots, a_n) \in A^n$ then we write $\mathbf{a} = a_1 \dots a_n$.

- (b) A *code of length n over A* is a nonempty subset of A^n . We refer to the elements of a code as *codewords* or *codevectors*.
- (c) If $|A| = m$ and C is a code over A then we call C an *m -ary code*. If $A = \{0, 1\}$ then C is a *binary code*. If $A = \{0, 1, 2\}$ then C is *ternary code*.
- (d) A code C is *trivial* if C has only one codeword. ▷

Example : The set $C_1 = \{0000, 1010, 0111, 1101\}$ is a binary code of length 4. ▷

2.2 Hamming Distance, Minimum Distance and Nearest Neighbor Decoding

Definition 2.2 Let A be an alphabet, $n \in \mathbb{N}$ and $\mathbf{x} = x_1 \dots x_n, \mathbf{y} = y_1 \dots y_n \in A^n$. The *Hamming distance between \mathbf{x} and \mathbf{y}* (notation : $d_H(\mathbf{x}, \mathbf{y})$ or $d(\mathbf{y})$) is the number of positions in which \mathbf{x} and \mathbf{y} differ. So

$$d_H(\mathbf{x}, \mathbf{y}) = |\{1 \leq i \leq n : x_i \neq y_i\}|$$

Proposition 2.3 (A^n, d_H) is a metric space. So for all $\mathbf{x}, \mathbf{y}, \mathbf{z} \in A^n$, we have

(a) $d_H(\mathbf{x}, \mathbf{y}) \geq 0$

(b) $d_H(\mathbf{x}, \mathbf{y}) = 0 \Leftrightarrow \mathbf{x} = \mathbf{y}$

(c) $d_H(\mathbf{x}, \mathbf{y}) = d_H(\mathbf{y}, \mathbf{x})$

(d) *Triangle Inequality* : $d_H(\mathbf{x}, \mathbf{z}) \leq d_H(\mathbf{x}, \mathbf{y}) + d_H(\mathbf{y}, \mathbf{z})$

Proof : (a), (b) and (c) are obvious. So we only prove the Triangle inequality. Let $\mathbf{x}, \mathbf{y}, \mathbf{z} \in A^n$. Put $\mathbf{x} = x_1 \dots x_n$, $\mathbf{y} = y_1 \dots y_n$ and $\mathbf{z} = z_1 \dots z_n$. Pick $1 \leq i \leq n$ with $x_i \neq z_i$. Then there are two possibilities :

- $x_i \neq y_i$
- $x_i = y_i$ and $y_i \neq z_i$

Hence

$$\{1 \leq i \leq n : x_i \neq z_i\} \subseteq \{1 \leq i \leq n : x_i \neq y_i\} \cup \{1 \leq i \leq n : y_i \neq z_i\}$$

So $d_H(\mathbf{x}, \mathbf{z}) \leq d_H(\mathbf{x}, \mathbf{y}) + d_H(\mathbf{y}, \mathbf{z})$. □

Next, we describe a very common decoding method.

Definition 2.4 [Nearest Neighbor Decoding] Let C be a code of length n over A . In this course, we will use the following decoding strategy :

Decode a received word \mathbf{y} as the codeword that is the closest to \mathbf{y} . So we decode $\mathbf{y} \in A^n$ as $\mathbf{c} \in C$ where $d_H(\mathbf{y}, \mathbf{c}) \leq d_H(\mathbf{y}, \mathbf{x})$ for all $\mathbf{x} \in C$.

This method of decoding is called *Nearest Neighbor Decoding*. We will abbreviate this as NND. This works fine as long as there is a unique codeword closest to the received word. If there is more than one codeword closest to the received word, we can either declare a decoding error (this method is called *incomplete decoding*) or we can somehow choose one of the codewords closest to the received word (*complete decoding*). ▷

Examples :

- (a) the binary code $C_1 = \{0000, 1010, 0111, 1101\}$

When we receive $\mathbf{y} = 0100$ we decode as 0000. When we receive 1000 we either declare a decoding error or decode as 0000 or 1010.

- (b) $C_2 = \{0000, 1011, 0121, 2022, 0212, 1102, 2201, 1220, 2110\}$

When we receive 1111 we decode as 1011. ▷

So far, we used a very naive approach to implement NND : when we receive a word \mathbf{y} , we go over every single codeword \mathbf{c} and calculate $d_H(\mathbf{y}, \mathbf{c})$. Later, we will see that certain codes have very efficient implementations of NND.

Next, we define a very important parameter of a code and show how it measures the error-correcting capabilities of a code.

Definition 2.5 Let C be a non-trivial code of length n over A . The *minimum distance of C* (notation : $d(C)$) is the minimum Hamming distance between two distinct codewords. So

$$d(C) = \min\{d_H(\mathbf{x}, \mathbf{y}) : \mathbf{x}, \mathbf{y} \in C \text{ and } \mathbf{x} \neq \mathbf{y}\} \quad \triangleright$$

Example : Consider the ternary code

$$C_2 = \{0000, 1011, 0121, 2022, 0212, 1102, 2201, 1220, 2110\}$$

Then $d(C_2) = 3$. \triangleright

Remark : Calculating the minimum distance of a code C might be quite time consuming : it is possible we have to calculate $\frac{|C|(|C| - 1)}{2}$ Hamming distances. Later, we will see more efficient methods of finding the minimum distance of certain codes. \triangleright

Definition 2.6

- (a) Let $n, M, d \in \mathbb{N}$. Then a code C is called an (n, M, d) -code if C has length n , $|C| = M$ and $d(C) = d$.
- (b) Let $n, d \in \mathbb{N}$. Then a code C is called an (n, d) -code if C has length n and $d(C) = d$. \triangleright

Example : The ternary code $C_2 = \{0000, 1011, 0121, 2022, 0212, 1102, 2201, 1220, 2110\}$ is an $(4, 9, 3)$ -code. \triangleright

Definition 2.7 Let $\mathbf{x} \in A^n$ and $r \in \mathbb{N}$.

- (1) The *sphere with center \mathbf{x} and radius r* (notation : $S_r(\mathbf{x})$) is the set of all words in A^n whose distance to \mathbf{x} is at most r . So

$$S_r(\mathbf{x}) = \{\mathbf{y} \in A^n : d_H(\mathbf{x}, \mathbf{y}) \leq r\}$$

- (2) The *volume* of the sphere $S_r(\mathbf{x})$ (notation : $|S_r(\mathbf{x})|$) is the number of words in $S_r(\mathbf{x})$. \triangleright

It turns out that the volume of a sphere does not depend on its center.

Proposition 2.8 Let A be an alphabet, $\mathbf{x} \in A^n$ and $r \in \mathbb{N}$. Then

$$|S_r(\mathbf{x})| = \sum_{i=0}^{\min\{r,n\}} \binom{n}{i} (|A| - 1)^i$$

for all $\mathbf{x} \in A^n$.

Proof : Let $\mathbf{x} \in A^n$. Put $k = \min\{r, n\}$. Then

$$S_r(\mathbf{x}) = \biguplus_{i=0}^k \{\mathbf{y} \in A^n : d_H(\mathbf{x}, \mathbf{y}) = i\}$$

For $i = 0, 1, \dots, k$, we have that

$$|\{\mathbf{y} \in A^n : d_H(\mathbf{x}, \mathbf{y}) = i\}| = \binom{n}{i} (|A| - 1)^i$$

Indeed, there are $\binom{n}{i}$ possibilities to choose i positions of \mathbf{y} for which $y_i \neq x_i$. For each of these i positions, there are $(|A| - 1)$ choices for y_i with $y_i \neq x_i$.

Hence

$$|S_r(\mathbf{x})| = \sum_{i=0}^k |\{\mathbf{y} \in A^n : d_H(\mathbf{x}, \mathbf{y}) = i\}| = \sum_{i=0}^k \binom{n}{i} (|A| - 1)^i \quad \square$$

The next proposition shows the relation between the error-correcting capabilities of a code and its minimum distance.

Proposition 2.9 Let C be a non-trivial code of length n over A and $e \in \mathbb{N}$. Then the following are equivalent :

- (a) Using NND, we will decode correctly if at most e errors occur during transmission.
- (b) $S_e(\mathbf{x}) \cap S_e(\mathbf{y}) = \emptyset$ for all distinct $\mathbf{x}, \mathbf{y} \in C$.
- (c) The minimum distance of C is at least $2e + 1$.

Proof : (a) \implies (b) Assume that we will decode correctly using NND if at most e errors occur during transmission. Let $\mathbf{x}, \mathbf{y} \in C$ with $S_e(\mathbf{x}) \cap S_e(\mathbf{y}) \neq \emptyset$. Let $\mathbf{z} \in S_e(\mathbf{x}) \cap S_e(\mathbf{y})$. View \mathbf{z} as a received word. On one hand, we should decode \mathbf{z} as \mathbf{x} since $d_H(\mathbf{z}, \mathbf{x}) \leq e$. On the other hand, we should decode \mathbf{z} as \mathbf{y} since $d_H(\mathbf{z}, \mathbf{y}) \leq e$. Since we decode correctly if at most e errors occur, it must be that $\mathbf{x} = \mathbf{y}$.

(b) Assume that $S_e(\mathbf{x}) \cap S_e(\mathbf{y}) = \emptyset$ for all distinct $\mathbf{x}, \mathbf{y} \in C$. Let $\mathbf{x}, \mathbf{y} \in C$ with $d(C) = d_H(\mathbf{x}, \mathbf{y})$. Put $k = \min\{d(C), e\}$. Choose $1 \leq i_1 < i_2 < \dots < i_k \leq n$ with $x_{i_j} \neq y_{i_j}$ for $j = 1, 2, \dots, k$. Put $\mathbf{z} = z_1 z_2 \dots z_n$ where for $i = 1, 2, \dots, n$, we have that $z_i = y_i$ if $i \notin \{i_1, i_2, \dots, i_k\}$ and $z_i = x_i$ if $i \in \{i_1, i_2, \dots, i_k\}$. Then $d_H(\mathbf{y}, \mathbf{z}) = k \leq e$. So $\mathbf{z} \in S_e(\mathbf{y})$. Hence $\mathbf{z} \notin S_e(\mathbf{x})$ since

$S_e(\mathbf{x}) \cap S_e(\mathbf{y}) = \emptyset$. So $d_H(\mathbf{x}, \mathbf{z}) > e$. If $d(C) \leq e$ then $\mathbf{z} = \mathbf{x}$, a contradiction. So $d(C) > e$. Then $e < d_H(\mathbf{x}, \mathbf{z}) = d(C) - e$. Hence $d(C) > 2e$. So $d(C) \geq 2e + 1$.

(c) Assume that $d(C) \geq 2e + 1$. Let $\mathbf{x} \in C$ and $\mathbf{y} \in A^n$ with $d_H(\mathbf{x}, \mathbf{y}) \leq e$. We apply NND on the received word \mathbf{y} . So we are looking for codewords whose distance to \mathbf{y} is minimal. Let $\mathbf{z} \in C$ with $d_H(\mathbf{y}, \mathbf{z}) \leq e$. Using the Triangle Inequality, we get that

$$d_H(\mathbf{x}, \mathbf{z}) \leq d_H(\mathbf{x}, \mathbf{y}) + d_H(\mathbf{y}, \mathbf{z}) \leq e + e = 2e < d(C)$$

Hence $\mathbf{z} = \mathbf{x}$. Thus \mathbf{x} is the unique codeword closest to \mathbf{y} . So using NND, we will decode \mathbf{y} correctly as \mathbf{x} . \square

Definition 2.10 A code C is a t -error correcting code if C satisfies any of the conditions in Proposition 2.9 and t is maximal with that property. So $t = \left\lfloor \frac{d(C) - 1}{2} \right\rfloor$. \triangleright

Example : The ternary code $C_2 = \{0000, 1011, 0121, 2022, 0212, 1102, 2201, 1220, 2110\}$ is a 1-error correcting code since $d(C_2) = 3$. \triangleright

2.3 Bounds

We begin this section with an upper bound on the size of an m -ary t -error correcting code of length n .

Theorem 2.11 (Sphere Packing Bound or Hamming Bound) *Let C be a t -error correcting m -ary code of length n . Then*

$$|C| \leq \frac{m^n}{\sum_{i=0}^t \binom{n}{i} (m-1)^i}$$

Proof : Since C is t -error correcting, it follows from Proposition 2.9(b) that $S_t(\mathbf{x}) \cap S_t(\mathbf{y}) = \emptyset$ for all distinct $\mathbf{x}, \mathbf{y} \in C$. Hence

$$\sum_{\mathbf{x} \in C} |S_t(\mathbf{x})| \leq |A^n|$$

where A is the alphabet of C . So $|A^n| = m^n$. By Proposition 2.8, we have that

$$V := |S_t(\mathbf{x})| = \sum_{i=0}^t \binom{n}{i} (m-1)^i$$

for all $\mathbf{x} \in C$. So

$$\sum_{\mathbf{x} \in C} |S_t(\mathbf{x})| = \sum_{\mathbf{x} \in C} V = |C|V$$

Since $|C|V \leq m^n$, we get that

$$|C| \leq \frac{m^n}{V} = \frac{m^n}{\sum_{i=0}^t \binom{n}{i} (m-1)^i} \quad \square$$

Example : Let C be a binary one-error correcting code of length 5. Then

$$|C| \leq \frac{2^5}{\binom{5}{0} + \binom{5}{1}} = \frac{32}{6} < 6$$

So $|C| \leq 5$. This does NOT imply that there exists such a code with five codewords. In fact, one can show that $|C| \leq 4$. This time, there exists a code with these parameters : $C = \{00000, 11110, 11001, 00111\}$ is an example. \triangleright

Definition 2.12

A code C is a *perfect* t -error correcting m -ary code of length n if there is equality in the Sphere Packing Bound, so if

$$|C| \left(\sum_{i=0}^t \binom{n}{i} (m-1)^i \right) = m^n \quad \triangleright$$

Example : Let $n \in \mathbb{N}$ be odd. Consider the code

$$C = \{ \underbrace{00 \dots 00}_{n \text{ times}}, \underbrace{11 \dots 11}_{n \text{ times}} \}$$

Then C is a perfect binary $\frac{n-1}{2}$ -error correcting code of length n . \triangleright

Remarks

- (a) The Sphere Packing Bound puts severe restrictions on the parameters of a t -error correcting m -ary code of length n . For example, there does not exist a perfect binary one-error correcting code of length 6 since

$$\frac{2^6}{\binom{6}{0} + \binom{6}{1}} = \frac{64}{7} \notin \mathbb{N}$$

- (b) If $\lambda, t, m, n \in \mathbb{N}$ such that $\lambda \left(\sum_{i=0}^t \binom{n}{i} (m-1)^i \right) = m^n$ then there may or may not exist a perfect t -error correcting m -ary code C of length n with $|C| = \lambda$. For example, there exists a perfect binary 3-error correcting code of length 23 but there does not exist a perfect binary 2-error correcting code of length 90 although $\frac{2^{90}}{\binom{90}{0} + \binom{90}{1} + \binom{90}{2}} \in \mathbb{N}$.

- (c) All 4-tuples (λ, t, m, n) such that there exist a perfect t -error correcting m -ary code C of length n with $|C| = \lambda$ are known. \triangleright

The next theorem guarantees the existence of an m -ary t -error correcting code of length n of a certain size.

Theorem 2.13 (Gilbert-Varshamov Bound) Let $m, t, n \in \mathbb{N}$ with $m \geq 2$ and $2t < n$. Then there exists an m -ary t -error correcting code C of length n with

$$|C| \geq \frac{m^n}{\sum_{i=0}^{2t} \binom{n}{i} (m-1)^i}$$

Proof : Let A be an alphabet of size m . Put $d = 2t + 1$. Since $n \geq d$, there exists two words $\mathbf{x}, \mathbf{y} \in A^n$ with $d_H(\mathbf{x}, \mathbf{y}) = d$. So there exists at least one (n, d) -code over A (namely $\{\mathbf{x}, \mathbf{y}\}$). Let C be an (n, d) -code over A with $|C|$ maximal. Put

$$V = \bigcup_{\mathbf{x} \in C} S_{2t}(\mathbf{x})$$

Suppose that $V \neq A^n$. Let $\mathbf{z} \in A^n \setminus V$. Put $C' = C \cup \{\mathbf{z}\}$. Since $\mathbf{z} \notin V$, we have that $d_H(\mathbf{z}, \mathbf{x}) > 2t$ and so $d_H(\mathbf{z}, \mathbf{x}) \geq d$ for all $\mathbf{x} \in C$. But $d(C) = d$. Hence $d(C') = d$, a contradiction to the maximality of $|C|$ since $|C'| = |C| + 1$.

So $V = A^n$. It follows from Proposition 2.8 that

$$m^n = |A^n| = |V| = \left| \bigcup_{\mathbf{x} \in C} S_{2t}(\mathbf{x}) \right| \leq \sum_{\mathbf{x} \in C} |S_{2t}(\mathbf{x})| = \sum_{\mathbf{x} \in C} \left(\sum_{i=0}^{2t} \binom{n}{i} (m-1)^i \right) = |C| \left(\sum_{i=0}^{2t} \binom{n}{i} (m-1)^i \right)$$

So

$$|C| \geq \frac{m^n}{\sum_{i=0}^{2t} \binom{n}{i} (m-1)^i} \quad \square$$

Example : Let $m = 3$, $t = 1$ and $n = 7$. Then there exists a ternary one-error correcting code of size 23 since

$$\frac{3^7}{\binom{7}{0} + \binom{7}{1} \cdot 2 + \binom{7}{2} \cdot 2^2} = \frac{2187}{99} \approx 22.09 \quad \triangleright$$

Chapter 3

Linear Codes

In this chapter, we consider codes that are vector spaces. Almost all codes we consider in this course are vector spaces.

3.1 Basic Definitions

Definition 3.1 Let \mathbb{F} be a finite field.

- (1) A code C of length n is a *linear code over \mathbb{F}* if C is a subspace of \mathbb{F}^n .
- (2) A code C is an $[n, k]$ -*code over \mathbb{F}* if C is a linear code of length n over \mathbb{F} and with minimum distance d .
- (3) A code C is an $[n, k, d]$ -*code over \mathbb{F}* if C is a linear code of length n over \mathbb{F} of dimension k and with minimum distance d . ▷

Example : $C = \{000, 110, 101, 011\}$ is a $[3, 2, 2]$ -code over $\mathbb{F}_2 = \{0, 1\}$. ▷

Remarks :

- (a) Let $C \subseteq \mathbb{F}^n$. Then C is a linear code over \mathbb{F} if and only if $\alpha\mathbf{x} + \beta\mathbf{y} \in C$ for all $\alpha, \beta \in \mathbb{F}$ and all $\mathbf{x}, \mathbf{y} \in C$.
- (b) Let C be an $[n, k]$ -code over \mathbb{F} . If $\{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_k\}$ is a basis for C over \mathbb{F} then

$$C = \{\alpha_1\mathbf{x}_1 + \alpha_2\mathbf{x}_2 + \dots + \alpha_k\mathbf{x}_k : \alpha_1, \alpha_2, \dots, \alpha_k \in \mathbb{F}\}$$

So $|C| = |\mathbb{F}|^k$. ▷

Example : $\{110, 101\}$ is a basis for $C = \{000, 110, 101, 011\}$ over $\{0, 1\}$. ▷

Definition 3.2 Let \mathbb{F} be a finite field.

- (a) For $\mathbf{x} \in \mathbb{F}^n$ we define the *Hamming weight of \mathbf{x}* (notation : $w_H(\mathbf{x})$ or $w(\mathbf{x})$) as the number of nonzero entries of \mathbf{x} . So if $\mathbf{x} = (x_1, \dots, x_n)$ then

$$w_H(\mathbf{x}) = |\{1 \leq i \leq n : x_i \neq 0\}|$$

- (b) Let C be a non-trivial linear code over \mathbb{F} . The *minimum weight of C* (notation : $w(C)$) is the minimum nonzero weight of a codeword. So $w(C) = \min\{w_H(\mathbf{x}) : \mathbf{0} \neq \mathbf{x} \in C\}$. \triangleright

Examples :

(a) $w(10110) = 3$

(b) Let $C = \{000, 110, 101, 011\}$. Then $w(C) = 2$. \triangleright

Remark : There is a connection between the Hamming distance and the Hamming weight. Let $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$. Then

$$d_H(\mathbf{x}, \mathbf{y}) = d_H(\mathbf{x} - \mathbf{y}, \mathbf{0}) = w(\mathbf{x} - \mathbf{y})$$

Indeed,

$$d_H(\mathbf{x}, \mathbf{y}) = |\{1 \leq i \leq n; x_i \neq y_i\}| = |\{1 \leq i \leq n : x_i - y_i \neq 0\}| = d_H(\mathbf{x} - \mathbf{y}, \mathbf{0}) = w(\mathbf{x} - \mathbf{y}) \quad \triangleright$$

The next proposition allows us to calculate the minimum distance of a linear code a little easier.

Proposition 3.3 Let C be a non-trivial linear code over \mathbb{F} . Then the minimum weight of C is equal to the minimum distance of C .

Proof : Let $\mathbf{x}, \mathbf{y} \in C$ with $d_H(\mathbf{x}, \mathbf{y}) = d(C)$. Note that $\mathbf{x} \neq \mathbf{y}$ and so $\mathbf{x} - \mathbf{y} \neq \mathbf{0}$. Hence $d(C) = d_H(\mathbf{x}, \mathbf{y}) = w(\mathbf{x} - \mathbf{y}) \geq w(C)$.

Let $\mathbf{z} \in C$ with $w(C) = w(\mathbf{z})$. Note that $\mathbf{z} \neq \mathbf{0}$. Hence $w(C) = w(\mathbf{z}) = d_H(\mathbf{z}, \mathbf{0}) \geq d(C)$.

So $w(C) = d(C)$. \square

Remark : To calculate the minimum weight/distance of a linear code C , we have to check at most $|C| - 1$ weights. Recall that we might have to check $\frac{|C|(|C| - 1)}{2}$ distances to calculate the minimum distance of a non-linear code C . Later, we will see an even more efficient method to calculate the minimum distance of a linear code using parity check matrices. \triangleright

3.2 Generator Matrix

Recall that every vector space of dimension k has a basis with k elements. For linear codes, we write a basis in matrix form.

Definition 3.4 Let C be an $[n, k]$ -code over \mathbb{F} . A matrix G is called a *generator matrix* for C if G is a $k \times n$ -matrix over \mathbb{F} whose rows form a basis for C over \mathbb{F} .

Example : Let $C = \{000, 110, 101, 011\}$. Then $\begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}$ is a generator matrix for C . \triangleright

Remark : A generator matrix provides us with an easy ‘encoder’ (a way of adding redundancy or to change an original message of k symbols into a codeword of n symbols) since C is the set of all linear combinations of rows of G :

$$C = \{ [x_1 \ \cdots \ x_k] G : x_1, \dots, x_k \in \mathbb{F} \}$$

For the previous example, we would get

message	codeword
00	000
10	110
01	011
11	101

\triangleright

Definition 3.5 Let C be an $[n, k]$ -code and G a generator matrix for C . Then G is in *standard form* if G is of the form $[I_k \ A]$ where I_k is the $k \times k$ identity matrix. \triangleright

Example : Let $C = \{000, 110, 101, 011\}$. Then $\begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$ is a generator matrix for C in standard form. \triangleright

Remark : A generator matrix in standard form provides us with a very easy ‘encoder’ and ‘decoder’ (a way of retrieving the original message of k symbols from the codeword of n symbols) : if we decode a received word as the codeword $x_1 \dots x_n$ then $x_1 \dots x_k$ was the original message. \triangleright

3.3 Parity Check Matrix

In this section, we define the very important concept of a parity check matrix.

Definition 3.6 Let C be an $[n, k]$ -code over \mathbb{F} . Then a matrix H is a *parity check matrix* for C if H is an $(n - k) \times n$ -matrix of rank $n - k$ over \mathbb{F} such that $\mathbf{x}H^T = \mathbf{0}$ for all $\mathbf{x} \in C$. \triangleright

Example : Let $C = \{000, 110, 101, 011\}$. Then $\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$ is a parity check matrix for C . \triangleright

Remark : If H is an $(n - k) \times n$ -matrix over \mathbb{F} and $\mathbf{x} \in \mathbb{F}^n$ then $\mathbf{x}H^T = \mathbf{0} \Leftrightarrow H\mathbf{x}^T = 0_{(n-k) \times 1}$.

A very common way of defining linear codes is not by using a generator matrix but by using a parity check matrix. Indeed, let H be an $(n - k) \times n$ -matrix over \mathbb{F} of rank $n - k$. Put $C = \{\mathbf{x} \in \mathbb{F}^n : \mathbf{x}H^T = \mathbf{0}\}$. Then it follows from standard linear algebra that C is an $[n, k]$ -code over \mathbb{F} and H is a parity check matrix for C . \triangleright

Example : Let $\mathbb{F} = \{0, 1\}$, $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 \end{bmatrix}$ and $C = \{\mathbf{x} \in \mathbb{F}^5 : \mathbf{x}H^T = \mathbf{0}\}$. Since

$$\mathbf{x}H^T = \mathbf{0} \Leftrightarrow \begin{cases} x_1 + x_4 = 0 \\ x_2 + x_5 = 0 \\ x_3 + x_4 + x_5 = 0 \end{cases}$$

we get that

$$C = \{00000, 10110, 01101, 11011\}$$

which is a binary $[5, 2, 3]$ -code. \triangleright

Given a generator matrix in standard form for a linear code C , we can easily write down a parity check matrix for C .

Proposition 3.7 *Let C be an $[n, k]$ -code over \mathbb{F} and $G = [I_k \ A]$ a generator matrix for C in standard form. Then $H := [-A^T \ I_{n-k}]$ is a parity check matrix for C .*

Proof : Clearly, H is an $(n - k) \times n$ -matrix over \mathbb{F} of rank $n - k$. So we only need to show that $\mathbf{x}H^T = \mathbf{0}$ for all $\mathbf{x} \in C$. Let $\mathbf{x} \in C$. Since G is a generator matrix for C , there exist $\lambda_1, \dots, \lambda_k \in \mathbb{F}$ such that $\mathbf{x} = [\lambda_1 \ \dots \ \lambda_k]G$. Hence

$$\mathbf{x}H^T = [\lambda_1 \ \dots \ \lambda_n]GH^T = [\lambda_1 \ \dots \ \lambda_n] [I_k \ A] [-A^T \ I_{n-k}]^T$$

Note that

$$[I_k \ A] [-A^T \ I_{n-k}]^T = [I_k \ A] \begin{bmatrix} -A \\ I_{n-k} \end{bmatrix} = -A + A = 0_{k \times (n-k)}$$

So $\mathbf{x}H^T = \mathbf{0}$. \square

The next proposition shows us how to use a parity check matrix of a linear code C to find the minimum distance of C .

Proposition 3.8 *Let C be an $[n, k]$ -code over \mathbb{F} , H a parity check matrix for C and d the minimum distance of C . Then every collection of $d - 1$ columns of H is linearly independent over \mathbb{F} and there is at least one collection of d columns of H that is linearly dependent over \mathbb{F} .*

Proof : Consider a collection of $d-1$ columns of H , say C_1, C_2, \dots, C_{d-1} . Let $\lambda_1, \lambda_2, \dots, \lambda_{d-1} \in \mathbb{F}$ with

$$\lambda_1 C_1 + \lambda_2 C_2 + \dots + \lambda_{d-1} C_{d-1} = \mathbf{0}_{(n-k) \times 1}$$

Put $\mathbf{x} = (\lambda_1, \lambda_2, \dots, \lambda_{d-1}, 0, \dots, 0) \in \mathbb{F}^n$. Then

$$\mathbf{x}H^T = \lambda_1 C_1 + \lambda_2 C_2 + \dots + \lambda_{d-1} C_{d-1} = \mathbf{0}_{(n-k) \times 1}$$

So $H\mathbf{x}^T = \mathbf{0}_{(n-k) \times 1}$ and $\mathbf{x} \in C$. If $\mathbf{x} \neq \mathbf{0}$ then $w(\mathbf{x}) \leq w(C) = d$, a contradiction since $w(\mathbf{x}) \leq d-1$. Hence $\mathbf{x} = \mathbf{0}$. So $\lambda_1 = \lambda_2 = \dots = \lambda_{d-1} = 0$. Thus every collection of $d-1$ columns of H is linearly independent over \mathbb{F} .

Let $\mathbf{x} \in C$ with $w(\mathbf{x}) = w(C) = d$. Then \mathbf{x} has d non-zero components, say in positions $1 \leq j_1 < j_2 < \dots < j_d \leq n$. Since $\mathbf{x}H^T = \mathbf{0}$, we get that

$$\mathbf{0}_{(n-k) \times 1} = H\mathbf{x}^T = x_1 C_1 + \dots + x_n C_n = x_{j_1} C_{j_1} + \dots + x_{j_d} C_{j_d}$$

Hence the collection $\{C_{j_1}, C_{j_2}, \dots, C_{j_d}\}$ of d columns of H is linearly dependent over \mathbb{F} . □

Example : Let C be the binary code of length 7 with the following parity check matrix :

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

Since H does not contain a zero-column, we get that every column in H is linearly independent over $\{0, 1\}$.

Since any two columns in H are distinct, we see that any two columns of H are linearly independent over $\{0, 1\}$.

But

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

So the first three columns of H are linearly dependent over $\{0, 1\}$.

Hence it follows from Proposition 3.8 that the minimum distance of C is three. ▷

Corollary 3.9 [*Singleton Bound*] Let C be an $[n, k, d]$ -code. Then $n - k \geq d - 1$.

Proof : Let H be a parity check matrix for C . Then the rank of H is $n - k$. By Proposition 3.8, H has $d - 1$ linearly independent columns. Hence $n - k \geq d - 1$. □

Definition 3.10 Let C be an $[n, k, d]$ -code. Then C is a *maximum distance separable code* (notation : MDS-code) if $n - k = d - 1$. ▷

Example Consider the ternary code C with parity check matrix $H = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \end{bmatrix}$. Then C is a $[4, 2, 3]$ -CODE. So C is a ternary MDS-code. ▷

3.4 Binary Hamming Codes

In this section, we define a special family of binary linear codes : the binary Hamming codes.

Definition 3.11 Let $r \in \mathbb{N}$. A *binary Hamming code with parameter r* is a binary code with a parity check matrix whose columns are all the non-zero vectors of length r . \triangleright

Example : For $r = 2$, we can put $H = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}$. Then the code

$$C := \left\{ x_1x_2x_3 \in \{0,1\}^3 : \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \\ 1 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 \end{bmatrix} \right\}$$

is a binary Hamming code with parameter r . \triangleright

The binary Hamming codes are examples of perfect codes.

Proposition 3.12 Let $r \in \mathbb{N}$. Then a binary Hamming code with parameter r is a perfect binary $[2^r - 1, 2^r - 1 - r, 3]$ -code.

Proof : Let H be a matrix whose columns are all the non-zero vectors of length r . Let C be the binary code with H as a parity check matrix. Note that there are $2^r - 1$ non-zero vectors of length r . So H is an $r \times (2^r - 1)$ -matrix. Since H contains I_r as a submatrix, we see that the rank of H is r . Hence C is a binary $[2^r - 1, 2^r - 1 - r]$ -code. Since H does not contain a zero column and since any two columns of H are distinct, we get that any two columns of H are linearly independent over $\{0,1\}$. But H contains the following columns :

$$\begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \end{bmatrix}^T, \quad \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 \end{bmatrix}^T \quad \text{and} \quad \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \end{bmatrix}^T$$

Since these three columns are linearly dependent over $\{0,1\}$, it follows from Proposition 3.8 that $d(C) = 3$.

By HW 2 Ex#4, any binary $[2^r - 1, 2^r - 1 - r, 3]$ -code is perfect. \square

We now describe a specific parity check matrix for a binary Hamming code. This parity check matrix comes in very handy when decoding binary Hamming codes.

Let $r \in \mathbb{N}$. We can view a binary vector of length r as the binary expansion of a natural number :

$$x_1x_2 \dots x_r \leftrightarrow x_r \cdot 2^0 + x_{r-1} \cdot 2^1 + x_{r-2} \cdot 2^2 + \cdots + x_2 \cdot 2^{r-2} + x_1 \cdot 2^{r-1}$$

So 110101 is the binary expansion of 53. We now order the columns of the parity check matrix such that the j -th column contains the binary expansion of j for $1 \leq j \leq 2^r - 1$. We denote the binary Hamming code corresponding to this parity check matrix by $Ham(2, r)$.

Examples : $\begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$ is a parity check matrix for $Ham(2, 2)$ and $\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$ is a parity check matrix for $Ham(2, 3)$. \triangleright

3.5 Cosets and Coset Leaders

Definition 3.13 Let C be an $[n, k]$ -code over \mathbb{F} .

- (a) A *coset of C* is a coset of the subgroup C of the group $(\mathbb{F}^n, +)$. So a coset of C is a set of the form

$$\mathbf{x} + C = \{\mathbf{x} + \mathbf{y} : \mathbf{y} \in C\}$$

where $\mathbf{x} \in \mathbb{F}^n$.

- (b) A *coset leader* of a coset is a vector of smallest weight in that coset. ▷

Recall from Abstract Algebra that the cosets of C form a partition of \mathbb{F}^n and that all cosets have the same size. So if C is an $[n, k]$ -code over \mathbb{F} then every coset of C contains $|\mathbb{F}|^k$ vectors and C has $|\mathbb{F}|^{n-k}$ cosets.

Example : Let C be the binary code with parity check matrix $\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$. Then the cosets of C are

$$\begin{aligned} C &= \{0000, 1010, 1101, 0111\} \\ C_1 &= \{1000, 0010, 0101, 1111\} \\ C_2 &= \{0100, 1110, 1001, 0011\} \\ C_3 &= \{0001, 1011, 1100, 0110\} \end{aligned}$$

We see that 0000 is the unique coset leader of C , 0100 is the unique coset leader of C_2 and 0001 is the unique coset leader of C_3 . But C_1 has two coset leaders : 1000 and 0010. ▷

3.6 Standard Array Decoding

In this section, we describe one (unpractical) implementation of Nearest Neighbor Decoding.

First, we introduce the concept of the *error vector*. Let C be an $[n, k, d]$ -code over \mathbb{F} . Suppose we transmit a codeword \mathbf{x} but receive the word \mathbf{y} . Then the *error vector* is $\mathbf{e} := \mathbf{y} - \mathbf{x}$. So $\mathbf{y} = \mathbf{x} + \mathbf{e}$. This implies that $d_H(\mathbf{y}, \mathbf{x}) = w(\mathbf{y} - \mathbf{x}) = w(\mathbf{e})$. When decoding \mathbf{y} , we are looking for a codeword \mathbf{c} such that $d_H(\mathbf{y}, \mathbf{c}) = w(\mathbf{y} - \mathbf{c})$ is minimal. Note that $\mathbf{y} - \mathbf{c}$ and \mathbf{y} belong to the same coset. Hence we get

We decode \mathbf{y} as the codeword \mathbf{c} if and only if $\mathbf{y} - \mathbf{c}$ is a coset leader of the coset containing \mathbf{y} . So the error vector is a coset leader of the coset containing \mathbf{y} .

Hence we follow this strategy for decoding \mathbf{y} :

- (1) Find a coset leader \mathbf{e} of the coset containing \mathbf{y} .
- (2) Decode \mathbf{y} as $\mathbf{y} - \mathbf{e}$.

We can implement this strategy using a *standard array*. This is a table with $|\mathbb{F}|^{n-k}$ rows and $|\mathbb{F}|^k$ columns.

- the first row contains all the codewords.
- the first column contains a coset leader for each coset.
- the word on the i -th row and j -column is the sum of the coset leader in position (i -th row, first column) and the codeword in position (first row, j -th column).

So the rows of a standard array are the cosets of C .

Using a standard array, we decode \mathbf{y} as the codeword in the first row and the same column as \mathbf{y} .

Example : Consider the binary code $C = \{0000, 1010, 1101, 0111\}$. A standard array for C is

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

So we decode 1110 as 1010. ▷

3.7 Syndrome Decoding

Decoding using a standard array is unpractical as the size of the array becomes very large.

Given the coset leaders, it is enough to find the row (=coset) to which \mathbf{y} belongs when decoding \mathbf{y} ; we do not need to know the exact position of \mathbf{y} in the array.

So if we could somehow decide which one of the coset leaders is the coset leader of the coset containing \mathbf{y} , we would not need a standard array anymore. That is exactly the idea behind syndrome decoding.

Definition 3.14 Let C be an $[n, k, d]$ -code over \mathbb{F} and H a parity check matrix for C . For $\mathbf{y} \in \mathbb{F}^n$, we define the *syndrome of \mathbf{y} (with respect to H)* (notation : $\text{syn}_H(\mathbf{y})$ or $\text{syn}(\mathbf{y})$) as

$$\text{syn}_H(\mathbf{y}) = H\mathbf{y}^T \quad \triangleright$$

Example : Consider the binary Hamming code $Ham(7, 4)$. For $\mathbf{y} = y_1y_2 \dots y_7 \in \{0, 1\}^7$, we get that

$$\text{syn}(\mathbf{y}) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} y_4 + y_5 + y_6 + y_7 \\ y_2 + y_3 + y_6 + y_7 \\ y_1 + y_3 + y_5 + y_7 \end{bmatrix}$$

So $\text{syn}(\mathbf{y})$ is a 3×1 -matrix. ▷

It turns out that every word in a coset has the same syndrome and different cosets have different syndromes.

Proposition 3.15 Let C be an $[n, k, d]$ -code over \mathbb{F} and H a parity check matrix for C . Then for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$, we have that $\text{syn}(\mathbf{x}) = \text{syn}(\mathbf{y})$ if and only if \mathbf{x} and \mathbf{y} belong to the same coset of C .

Proof : We get that

$$\begin{aligned} \text{syn}(\mathbf{x}) = \text{syn}(\mathbf{y}) &\Leftrightarrow H\mathbf{x}^T = H\mathbf{y}^T \\ &\Leftrightarrow H(\mathbf{y} - \mathbf{x})^T = 0_{(n-k) \times 1} \\ &\Leftrightarrow \mathbf{y} - \mathbf{x} \in C \\ &\Leftrightarrow \mathbf{y} - \mathbf{x} = \mathbf{c} \text{ for some } \mathbf{c} \in C \\ &\Leftrightarrow \mathbf{y} = \mathbf{x} + \mathbf{c} \text{ for some } \mathbf{c} \in C \\ &\Leftrightarrow \mathbf{y} \in \mathbf{x} + C \end{aligned}$$

So $\text{syn}(\mathbf{x}) = \text{syn}(\mathbf{y})$ if and only if \mathbf{y} belongs to the coset of \mathbf{x} . □

Example : Let C be the binary code with parity check matrix $\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$. Then the syndromes are

$$\begin{aligned} \text{syn}(0000) &= \text{syn}(1010) = \text{syn}(1101) = \text{syn}(0111) = [0 \ 0]^T \\ \text{syn}(1000) &= \text{syn}(0010) = \text{syn}(0101) = \text{syn}(1111) = [1 \ 0]^T \\ \text{syn}(0100) &= \text{syn}(1110) = \text{syn}(1001) = \text{syn}(0011) = [0 \ 1]^T \\ \text{syn}(0001) &= \text{syn}(1011) = \text{syn}(1100) = \text{syn}(0110) = [1 \ 1]^T \end{aligned} \quad \triangleright$$

This means that we can use a *syndrome table* instead of a standard array. For each coset, we list a coset leader and its syndrome. When we want to decode \mathbf{y} , we calculate $\text{syn}(\mathbf{y})$ and look it up in the syndrome table. We have that $\text{syn}(\mathbf{y}) = \text{syn}(\mathbf{e})$ for exactly one coset leader in the table. We decode \mathbf{y} as $\mathbf{y} - \mathbf{e}$.

Example : Let C be the binary code with parity check matrix $\begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$. A syndrome table for C is

coset leader	syndrome
0000	$[0 \ 0]^T$
1000	$[1 \ 0]^T$
0100	$[0 \ 1]^T$
0001	$[1 \ 1]^T$

Suppose we receive $\mathbf{y} = 1110$. We easily find that

$$\text{syn}(\mathbf{y}) = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

From the syndrome table, we get that

$$\text{syn}(\mathbf{y}) = \text{syn}(0100)$$

Hence we decode \mathbf{y} as $\mathbf{y} - 0100 = 1110 - 0100 = 1010$. ▷

For the binary Hamming code $Ham(2, r)$, we do not need a syndrome table. Since $Ham(2, r)$ has 2^r cosets and has minimum distance three, we know all the coset leaders : any binary vector of length $2^r - 1$ and weight at most one. Moreover, if \mathbf{x} is a binary vector of length $2^r - 1$ and weight one (so \mathbf{x} has a one in exactly one position, say $x_j = 1$), then $\text{syn}(\mathbf{x})$ is the j -th column of the parity check matrix. But the j -th column of this parity check matrix is the binary expansion of the integer j . Hence we get the following syndrome decoding algorithm for $Ham(2, r)$:

- After receiving the word \mathbf{y} , calculate $\text{syn}(\mathbf{y})$.
- If $\text{syn}(\mathbf{y}) = \mathbf{0}$ then \mathbf{y} is a codeword and we decode \mathbf{y} as \mathbf{y} .
- If $\text{syn}(\mathbf{y}) \neq \mathbf{0}$ then we view $\text{syn}(\mathbf{y})$ as the binary expansion of the integer j (where $1 \leq j \leq 2^r - 1$). We decode \mathbf{y} by changing the j -th digit in \mathbf{y} .

Note that we will decode correctly IF at most one error occurred during transmission.

Example : Consider the binary Hamming code $Ham(2, 3)$ with parity check matrix

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

Suppose we receive the word $\mathbf{y} = 1110110$. We easily get

$$\text{syn}(\mathbf{y}) = \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

Since 011 is the binary expansion of 3, we assume that an error occurred in the third position. So we decode 1110110 as 1100110. ▷

3.8 Dual Codes

Recall that if $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$ then the dot product of \mathbf{x} and \mathbf{y} is

$$\mathbf{x} \cdot \mathbf{y} = x_1y_1 + x_2y_2 + \cdots + x_ny_n$$

If we think of \mathbf{x} and \mathbf{y} as $1 \times n$ -matrices then

$$\mathbf{x} \cdot \mathbf{y} = \mathbf{xy}^T = \mathbf{yx}^T$$

We say that \mathbf{x} and \mathbf{y} are *orthogonal* if $\mathbf{x} \cdot \mathbf{y} = 0$.

Definition 3.16 Let C be an $[n, k]$ -code over \mathbb{F} . Then the *dual code of C* (notation : C^\perp) is the code

$$C^\perp = \{\mathbf{x} \in \mathbb{F}^n : \mathbf{x} \cdot \mathbf{c} = 0 \text{ for all } \mathbf{c} \in C\} \quad \triangleright$$

If G is a generator matrix for C then every codeword is a linear combination of the rows of G . Hence the dual code C^\perp is

$$C^\perp = \{\mathbf{x} \in \mathbb{F}^n : G\mathbf{x}^T = \mathbf{0}_{k \times 1}\}$$

Example : Let C be the binary code with generator matrix

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

So $C = \{00000, 10010, 00101, 11011, 10111, 01001, 11110, 01100\}$. Then

$$x_1x_2x_3x_4x_5 \in C^\perp \Leftrightarrow \begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

So $C^\perp = \{00000, 10010, 01101, 11111\}$. \triangleright

The following theorem is proven in any undergraduate linear algebra class.

Theorem 3.17 Let C be an $[n, k]$ -code over \mathbb{F} . Then the following holds:

- (a) C^\perp is an $[n, n - k]$ -code over \mathbb{F} .
- (b) $C^{\perp\perp} = C$.

Proof : \square

This means that a generator matrix for C is a parity check matrix for C^\perp and a parity check matrix for C is a generator matrix for C^\perp .

Definition 3.18 Let C be an $[n, k]$ -code over \mathbb{F} . Then C is *self-dual* if $C = C^\perp$. ▷

Example : Let C be the binary code with generator matrix

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

Note that every row in G has weight four. So any row in G is orthogonal to itself. One easily checks that any two rows in G are orthogonal to each other. Since any codeword is a linear combination of rows in G , we get that $\mathbf{x} \cdot \mathbf{y} = 0$ for all $\mathbf{x}, \mathbf{y} \in C$. So $C \subseteq C^\perp$. But $\dim C = \dim C^\perp = 4$. So $C = C^\perp$. ▷

3.9 Extended Codes

Definition 3.19 Let \mathbb{F} be a field and $n \in \mathbb{N}$.

- (a) Given a word $\mathbf{x} \in \mathbb{F}^n$, we *extend* \mathbf{x} to a word $\mathbf{x}' \in \mathbb{F}^{n+1}$ of length $n + 1$ by adding a digit to \mathbf{x} in the following way :

$$\mathbf{x}' = x_1 \dots x_n x_{n+1} \Leftrightarrow \mathbf{x} = x_1 \dots x_n \text{ and } x_1 + x_2 + \dots + x_n + x_{n+1} = 0$$

- (b) Let C be an $[n, k]$ -code over \mathbb{F} . The *extended code* C' is the code of length $n + 1$ we get by extending every codeword in C . ▷

Example : Let C be the binary code $\{0000, 1110, 0111, 1001\}$. Then the extended code C' is $\{00000, 11101, 01111, 10010\}$. ▷

Proposition 3.20 Let C be an $[n, k, d]$ -code over \mathbb{F} . Then the following holds:

- (a) The extended code C' is an $[n + 1, k, d']$ -code over \mathbb{F} where $d' \in \{d, d + 1\}$. If $\mathbb{F} = \{0, 1\}$ then d' is even.
- (b) If G is a generator matrix for C then we get a generator matrix for C' by extending every row of G .
- (c) If H is a parity check matrix for C then $\begin{bmatrix} H & 0_{(n-k) \times 1} \\ 1 & \dots & 1 & 1 \end{bmatrix}$ is a parity check matrix for C' .

Proof : (a)(b) Clearly C' has length $n + 1$.

If $\mathbf{x} \in \mathbb{F}^n$ then we denote by (\mathbf{x}, x') the extended word in \mathbb{F}^{n+1} related to \mathbf{x} (so $\mathbf{x} = x_1 \dots x_n \in \mathbb{F}^n$ and $x' \in \mathbb{F}$ such that $x_1 + \dots + x_n + x' = 0$).

Let $\mathbf{v}_1, \dots, \mathbf{v}_k$ be a basis for C over \mathbb{F} . We will prove that $(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)$ is a basis for C' over \mathbb{F} .

First, we show that $C' = \text{span}\{(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)\}$.

Let $(\mathbf{x}, x') \in C'$. Then $\mathbf{x} \in C$ and so $\mathbf{x} = a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k$ for some $a_1, \dots, a_k \in \mathbb{F}$ since $\mathbf{v}_1, \dots, \mathbf{v}_k$ is a basis for C over \mathbb{F} . For $1 \leq i \leq k$ and $1 \leq j \leq n$, we denote the j -th component of \mathbf{v}_i by v_{ij} . Then

$$\begin{aligned} x' &= -x_1 - \dots - x_n \\ &= -(a_1v_{11} + \dots + a_kv_{k1}) - \dots - (a_1v_{1n} + \dots + a_kv_{kn}) \\ &= a_1(-v_{11} - \dots - v_{1n}) + \dots + a_k(-v_{k1} - \dots - v_{kn}) \\ &= a_1v'_1 + \dots + a_kv'_k \end{aligned}$$

Hence we get that

$$(\mathbf{x}, x') = (a_1\mathbf{v}_1 + \dots + a_k\mathbf{v}_k, a_1v'_1 + \dots + a_kv'_k) = a_1(\mathbf{v}_1, v'_1) + \dots + a_k(\mathbf{v}_k, v'_k)$$

So $C' \subseteq \text{span}\{(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)\}$.

Let $\mathbf{z} \in \text{span}\{(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)\}$. Then

$$\mathbf{z} = b_1(\mathbf{v}_1, v'_1) + \dots + b_k(\mathbf{v}_k, v'_k) = (b_1\mathbf{v}_1 + \dots + b_k\mathbf{v}_k, b_1v'_1 + \dots + b_kv'_k)$$

for some $b_1, \dots, b_k \in \mathbb{F}$. Put $\mathbf{y} = b_1\mathbf{v}_1 + \dots + b_k\mathbf{v}_k$. Then $\mathbf{y} \in C$. Similarly as above, we get that $b_1v'_1 + \dots + b_kv'_k = y'$. Hence

$$\mathbf{z} = b_1(\mathbf{v}_1, v'_1) + \dots + b_k(\mathbf{v}_k, v'_k) = (\mathbf{y}, y') \in C'$$

So $\text{span}\{(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)\} \subseteq C'$. Hence $C' = \text{span}\{(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)\}$.

Next, we show that $(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)$ are linearly independent over \mathbb{F} . Let $c_1, \dots, c_k \in \mathbb{F}$ with

$$c_1(\mathbf{v}_1, v'_1) + \dots + c_k(\mathbf{v}_k, v'_k) = \mathbf{0} \in \mathbb{F}^{n+1}$$

So

$$(c_1\mathbf{v}_1 + \dots + c_k\mathbf{v}_k, c_1v'_1 + \dots + c_kv'_k) = (\mathbf{0}, 0)$$

In particular, $c_1\mathbf{v}_1 + \dots + c_k\mathbf{v}_k = \mathbf{0} \in \mathbb{F}^n$. Since $\mathbf{v}_1, \dots, \mathbf{v}_k$ is a basis for C over \mathbb{F} , we have that $\mathbf{v}_1, \dots, \mathbf{v}_k$ are linearly independent over \mathbb{F} . Hence $c_1 = \dots = c_k = 0$. So $(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)$ are linearly independent over \mathbb{F} .

Hence $\{(\mathbf{v}_1, v'_1), \dots, (\mathbf{v}_k, v'_k)\}$ is a basis for C' over \mathbb{F} . So C' is an $[n, k]$ -code over \mathbb{F} . Note that this also shows that we get a generator matrix for C' by extending the rows of a generator matrix for C .

Finally, we show that $d(C') \in \{d(C), d(C) + 1\}$. Let $\mathbf{x} \in C$ with $w(\mathbf{x}) = d(C)$. Then $(\mathbf{x}, x') \in C'$ and so $d(C') \leq w(\mathbf{x}, x') \leq w(\mathbf{x}) + 1 = d(C) + 1$. Let $(\mathbf{y}, y') \in C'$ with

$w(\mathbf{y}, y') = d(C')$. If $\mathbf{y} = \mathbf{0} \in \mathbb{F}^n$ then $y' = 0$ and so $(\mathbf{y}, y') = \mathbf{0} \in \mathbb{F}^{n+1}$, a contradiction. So $\mathbf{y} \neq \mathbf{0}$. Hence $d(C') = w(\mathbf{y}, y') \geq w(\mathbf{y}) \geq d(C)$. Thus $d(C) \leq d(C') \leq d(C) + 1$. So $d(C') \in \{d(C), d(C) + 1\}$.

Suppose now that $\mathbb{F} = \{0, 1\}$ (so C is a binary code). If $x_1x_2 \dots x_nx_{n+1}$ is an extended word then $x_1 + x_2 + \dots + x_n + x_{n+1} = 0$. So the word $x_1x_2 \dots x_nx_{n+1}$ contains an even number of ones. Hence $w(\mathbf{z})$ is even for all $\mathbf{z} \in C'$. So $d(C')$ is even.

(c) Let H be a parity check matrix for C . Put $H' = \begin{bmatrix} H & 0_{(n-k) \times 1} \\ 1 & \dots & 1 & 1 \end{bmatrix}$. Let $\mathbf{x} \in C$. Put $x' = -x_1 - \dots - x_n$. Then $(x_1, \dots, x_n, x') = (\mathbf{x}, x') \in C'$ and

$$H'(\mathbf{x}, x')^T = \begin{bmatrix} H & 0_{(n-k) \times 1} \\ 1 & \dots & 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_n \\ x' \end{bmatrix} = \begin{bmatrix} H \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} \\ x_1 + \dots + x_n + x' \end{bmatrix} = \begin{bmatrix} 0_{(n-k) \times 1} \\ 0 \end{bmatrix}$$

since $H \begin{bmatrix} x_1 \\ \vdots \\ x_n \end{bmatrix} = H\mathbf{x}^T = 0_{(n-k) \times 1}$ (because $\mathbf{x} \in C$ and H is a parity check matrix for C) and $x_1 + \dots + x_n + x' = 0$.

So we showed that $H'\mathbf{z}^T = 0_{(n+1-k) \times 1}$ for all $\mathbf{z} \in C'$. Since the rank of H' is indeed $n + 1 - k$, we get that H' is a parity check matrix for C' . \square

Example : Let C be the binary code $\{0000, 1110, 0111, 1001\}$. Then $G = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$ is a generator matrix for C and so $H = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix}$ is a parity check matrix for C .

The extended code C' is $\{00000, 11101, 01111, 10010\}$. Note that $\begin{bmatrix} 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{bmatrix}$ is a generator matrix for C' and $\begin{bmatrix} 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ is a parity check matrix for C' . \triangleright

Chapter 4

Reed-Muller Codes

4.1 The $C_1 \otimes C_2$ -Construction

Definition 4.1 Let \mathbb{F} be a field, C_1 an $[n, k_1, d_1]$ -code over \mathbb{F} and C_2 an $[n, k_2, d_2]$ -code over \mathbb{F} . We define a new code $C_1 \otimes C_2$ of length $2n$ over \mathbb{F} as

$$C_1 \otimes C_2 = \{(\mathbf{x}, \mathbf{x} + \mathbf{y}) : \mathbf{x} \in C_1 \text{ and } \mathbf{y} \in C_2\} \quad \triangleright$$

Example : Let $C_1 = \{000, 110, 101, 011\}$ and $C_2 = \{000, 111\}$. Then

$$C_1 \otimes C_2 = \{000000, 110110, 101101, 011011, 000111, 110001, 101010, 011100\}. \quad \triangleright$$

Theorem 4.2 Let \mathbb{F} be a field and C_i an $[n, k_i, d_i]$ -code over \mathbb{F} for $i = 1, 2$. Then $C_1 \otimes C_2$ is an $[2n, k_1 + k_2, \min\{2d_1, d_2\}]$ -code over \mathbb{F} . Moreover, if G_i is a generator matrix for C_i for $i = 1, 2$, then $\begin{bmatrix} 0_{k_2 \times n} & G_2 \\ G_1 & G_1 \end{bmatrix}$ is a generator matrix for $C_1 \otimes C_2$.

Proof : Put $C = C_1 \otimes C_2$. Clearly, C is of length $2n$.

First, we prove that C is linear. So let $\mathbf{x}_1, \mathbf{x}_2 \in C_1$, $\mathbf{y}_1, \mathbf{y}_2 \in C_2$ and $a, b \in \mathbb{F}$. Then

$$a(\mathbf{x}_1, \mathbf{x}_1 + \mathbf{y}_1) + b(\mathbf{x}_2, \mathbf{x}_2 + \mathbf{y}_2) = (a\mathbf{x}_1 + b\mathbf{x}_2, a\mathbf{x}_1 + b\mathbf{x}_2 + a\mathbf{y}_1 + b\mathbf{y}_2)$$

Since C_1 and C_2 are linear, we get that $a\mathbf{x}_1 + b\mathbf{x}_2 \in C_1$ and $a\mathbf{y}_1 + b\mathbf{y}_2 \in C_2$. Hence

$$a(\mathbf{x}_1, \mathbf{x}_1 + \mathbf{y}_1) + b(\mathbf{x}_2, \mathbf{x}_2 + \mathbf{y}_2) \in C$$

So C is linear.

Next, we prove that $\dim(C) = k_1 + k_2$. Let $\{\mathbf{x}_1, \dots, \mathbf{x}_{k_1}\}$ be a basis for C_1 and $\{\mathbf{y}_1, \dots, \mathbf{y}_{k_2}\}$ a basis for C_2 . We will show that

$$\beta := \{(\mathbf{x}_1, \mathbf{x}_1), \dots, (\mathbf{x}_{k_1}, \mathbf{x}_{k_1}), (\mathbf{0}, \mathbf{y}_1), \dots, (\mathbf{0}, \mathbf{y}_{k_2})\}$$

is a basis for C . Note that $\beta \subseteq C$. So $\text{span}\beta \subseteq C$. Pick $\mathbf{x} \in C_1$ and $\mathbf{y} \in C_2$. Then $\mathbf{x} = a_1\mathbf{x}_1 + \cdots + a_{k_1}\mathbf{x}_{k_1}$ and $\mathbf{y} = b_1\mathbf{y}_1 + \cdots + b_{k_2}\mathbf{y}_{k_2}$ for some $a_1, \dots, a_{k_1}, b_1, \dots, b_{k_2} \in \mathbb{F}$. Hence

$$\begin{aligned} (\mathbf{x}, \mathbf{x} + \mathbf{y}) &= (a_1\mathbf{x}_1 + \cdots + a_{k_1}\mathbf{x}_{k_1}, a_1\mathbf{x}_1 + \cdots + a_{k_1}\mathbf{x}_{k_1} + b_1\mathbf{y}_1 + \cdots + b_{k_2}\mathbf{y}_{k_2}) \\ &= a_1(\mathbf{x}_1, \mathbf{x}_1) + \cdots + a_{k_1}(\mathbf{x}_{k_1}, \mathbf{x}_{k_1}) + b_1(\mathbf{0}, \mathbf{y}_1) + \cdots + b_{k_2}(\mathbf{0}, \mathbf{y}_{k_2}) \end{aligned}$$

So $C \subseteq \text{span}\beta$. Hence $C = \text{span}\beta$.

Let $\lambda_1, \dots, \lambda_{k_1}, \mu_1, \dots, \mu_{k_2} \in \mathbb{F}$ such that

$$\lambda_1(\mathbf{x}_1, \mathbf{x}_1) + \cdots + \lambda_{k_1}(\mathbf{x}_{k_1}, \mathbf{x}_{k_1}) + \mu_1(\mathbf{0}, \mathbf{y}_1) + \cdots + \mu_{k_2}(\mathbf{0}, \mathbf{y}_{k_2}) = \mathbf{0} \in \mathbb{F}^{2n}$$

Then

$$(\lambda_1\mathbf{x}_1 + \cdots + \lambda_{k_1}\mathbf{x}_{k_1}, \lambda_1\mathbf{x}_1 + \cdots + \lambda_{k_1}\mathbf{x}_{k_1} + \mu_1\mathbf{y}_1 + \cdots + \mu_{k_2}\mathbf{y}_{k_2}) = \mathbf{0} \in \mathbb{F}^{2n}$$

So

$$\lambda_1\mathbf{x}_1 + \cdots + \lambda_{k_1}\mathbf{x}_{k_1} = \mathbf{0} \in \mathbb{F}^n \quad (1)$$

and

$$\lambda_1\mathbf{x}_1 + \cdots + \lambda_{k_1}\mathbf{x}_{k_1} + \mu_1\mathbf{y}_1 + \cdots + \mu_{k_2}\mathbf{y}_{k_2} = \mathbf{0} \in \mathbb{F}^n \quad (2)$$

Since $\{\mathbf{x}_1, \dots, \mathbf{x}_{k_1}\}$ is a basis for C_1 , it follows from (1) that $\lambda_1 = \cdots = \lambda_{k_1} = 0$. Then we get from (2) that

$$\mu_1\mathbf{y}_1 + \cdots + \mu_{k_2}\mathbf{y}_{k_2} = \mathbf{0} \in \mathbb{F}^n$$

Since $\{\mathbf{y}_1, \dots, \mathbf{y}_{k_2}\}$ is a basis for C_2 , we see that $\mu_1, \dots, \mu_{k_2} = 0$. Hence β is linearly independent over \mathbb{F} . So β is a basis for C .

The statement about generator matrices now follows since the rows of a generator matrix for a code form a basis for that code.

Finally, we prove that $d(C) = \min\{2d_1, d_2\}$.

Let $\mathbf{u} \in C_1$ with $w(\mathbf{u}) = d_1$. Then $(\mathbf{u}, \mathbf{u}) \in C$ and so $2d_1 = w(\mathbf{u}, \mathbf{u}) \geq d(C)$. Let $\mathbf{v} \in C_2$ with $w(\mathbf{v}) = d_2$. Then $(\mathbf{0}, \mathbf{v}) \in C$ and so $d_2 = w(\mathbf{0}, \mathbf{v}) \geq d(C)$. So $d(C) \leq \min\{2d_1, d_2\}$.

Let $\mathbf{x} \in C_1$ and $\mathbf{y} \in C_2$ with $w(\mathbf{x}, \mathbf{x} + \mathbf{y}) = d(C)$. If $\mathbf{y} = \mathbf{0}$ then $\mathbf{x} \neq \mathbf{0}$ and so

$$d(C) = w(\mathbf{x}, \mathbf{x}) = 2w(\mathbf{x}) \geq 2d_1 \geq \min\{2d_1, d_2\}$$

If $\mathbf{y} \neq \mathbf{0}$ then

$$d(C) = w(\mathbf{x}, \mathbf{x} + \mathbf{y}) = w(\mathbf{x}) + w(\mathbf{x} + \mathbf{y}) \geq w(\mathbf{x}) + w(\mathbf{y}) - w(\mathbf{x}) = w(\mathbf{y}) \geq d_2 \geq \min\{2d_1, d_2\}$$

Either way, we have that $d(C) \geq \min\{2d_1, d_2\}$. So $d(C) = \min\{2d_1, d_2\}$. \square

4.2 Definition of the Reed-Muller Codes

We use the $C_1 \otimes C_2$ -construction to define the Reed-Muller codes.

Definition 4.3 [Reed-Muller Codes] Let $0 \leq r \leq m$. Then the r -th order Reed-Muller code of length 2^m (notation : $R(r, m)$) is a binary code defined as :

$$(a) R(0, m) = \{00 \dots 00, 11 \dots 11\} \subset \{0, 1\}^{2^m}$$

$$(b) R(m, m) = \{0, 1\}^{2^m}$$

$$(c) \text{ If } 1 \leq r \leq m - 1 \text{ then } R(r, m) = R(r, m - 1) \otimes R(r - 1, m - 1). \quad \triangleright$$

Examples : Note that $G(0, m) := [1 \ \dots \ 1]$ (a 1×2^m -matrix) is a generator matrix for $R(0, m)$ and $G(1, 1) := \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}$ is a generator matrix for $R(1, 1)$. Using Theorem 4.2, we can write down a generator matrix $G(r, m)$ for $R(r, m)$. We illustrate this for some first order Reed-Muller codes :

$$G(1, 2) = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 \end{bmatrix} \quad G(1, 3) = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

$$G(1, 4) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \triangleright$$

The dimension and minimum distance of the Reed-Muller codes are known.

Proposition 4.4 *Let $0 \leq r \leq m$. Then the dimension of $R(r, m)$ is $\binom{m}{0} + \binom{m}{1} + \dots + \binom{m}{r-1} + \binom{m}{r}$ and the minimum distance of $R(r, m)$ is 2^{m-r} .*

Proof : We proof this using induction on $m + r$.

Suppose first that $m + r = 0$. Then $m = r = 0$. Note that $R(0, 0) = \{0, 1\}$. So the dimension of $R(0, 0) = 1 = \binom{0}{0}$ and the minimum distance of $R(0, 0) = 1 = 2^{0-0}$.

Suppose next that the proposition holds if $m + r \leq n$ for some $n \geq 0$. Let $0 \leq r \leq m$ with $m + r = n + 1$. If $r = 0$ then $R(r, m) = \{00 \dots 00, 11 \dots 11\}$; so the dimension of $R(r, m)$ is $1 = \binom{m}{0}$ and the minimum distance of $R(r, m)$ is $2^m = 2^{m-r}$. If $r = m$ then $R(r, m) = \{0, 1\}^{2^m}$; so the dimension of $R(r, m)$ is $2^m = \binom{m}{0} + \binom{m}{1} + \dots + \binom{m}{m-1} + \binom{m}{m} = \binom{m}{0} + \dots + \binom{m}{r}$ by Newton's Binomium and the minimum distance of $R(r, m)$ is $1 = 2^{m-m} = 2^{m-r}$. So we may assume that $1 \leq r \leq m - 1$. Then $R(r, m) = R(r, m - 1) \otimes R(r - 1, m - 1)$. Using Theorem 4.2 and induction, we get that

$$\begin{aligned} \dim R(r, m) &= \dim R(r, m - 1) + \dim R(r - 1, m - 1) \\ &= \left\{ \binom{m-1}{0} + \dots + \binom{m-1}{r} \right\} + \left\{ \binom{m-1}{0} + \dots + \binom{m-1}{r-1} \right\} \\ &= \binom{m-1}{0} + \left\{ \binom{m-1}{1} + \binom{m-1}{0} \right\} + \left\{ \binom{m-1}{2} + \binom{m-1}{1} \right\} + \dots + \left\{ \binom{m-1}{r} + \binom{m-1}{r-1} \right\} \\ &= \binom{m}{0} + \binom{m}{1} + \binom{m}{2} + \dots + \binom{m}{r} \end{aligned}$$

since $\binom{m-1}{i} + \binom{m-1}{i-1} = \binom{m}{i}$ for $1 \leq i \leq r$.

Again using induction and Theorem 4.2, we find that

$$d(R(r, m)) = \min\{2d(R(r, m-1)), d(R(r-1, m-1))\} = \min\{2 \cdot 2^{m-1-r}, 2^{m-r}\} = 2^{m-r} \quad \square$$

Remark : the Reed-Muller code $R(r, m)$ can correct up to $2^{m-r-1} - 1$ errors. \triangleright

4.3 Majority Logic Decoding for Reed-Muller Codes

In this section, we illustrate *Majority Logic Decoding*, a very efficient decoding algorithm for Reed-Muller codes.

Syndrome decoding can be very impractical. Indeed, for $R(1, 5)$ (a code used by NASA in the seventies) a syndrome table would have $2^{26} = 67, 108, 864$ rows!

Consider the first-order Reed-Muller code $R(1, 4)$. This code has dimension 5 and minimum distance 8. So $R(1, 4)$ is three-error correcting. Let $G(1, 4)$ be the generator matrix we described in the examples on page 27. Then a codeword in $R(1, 4)$ is a linear combination of the rows of $G(1, 4)$. The decoding algorithm we describe will find this linear combination (so five symbols) instead of decoding to a codeword directly (sixteen symbols).

Let $x_1x_2 \dots x_{15}x_{16} \in R(1, 4)$. Then there exist $a_0, a_1, a_2, a_3, a_4 \in \{0, 1\}$ such that

$$x_1x_2 \dots x_{15}x_{16} = \begin{bmatrix} a_4 & a_3 & a_2 & a_1 & a_0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

So

$$\left\{ \begin{array}{l} x_1 = a_0 \\ x_2 = a_1 + a_0 \\ x_3 = a_2 + a_0 \\ x_4 = a_2 + a_1 + a_0 \\ x_5 = a_3 + a_0 \\ x_6 = a_3 + a_1 + a_0 \\ x_7 = a_3 + a_2 + a_0 \\ x_8 = a_3 + a_2 + a_1 + a_0 \\ x_9 = a_4 + a_0 \\ x_{10} = a_4 + a_1 + a_0 \\ x_{11} = a_4 + a_2 + a_0 \\ x_{12} = a_4 + a_2 + a_1 + a_0 \\ x_{13} = a_4 + a_3 + a_0 \\ x_{14} = a_4 + a_3 + a_1 + a_0 \\ x_{15} = a_4 + a_3 + a_2 + a_0 \\ x_{16} = a_4 + a_3 + a_2 + a_1 + a_0 \end{array} \right. \quad (*)$$

This leads to the following *check sums* for a_1, a_2, a_3 and a_4 :

$$\begin{aligned} a_1 &= x_1 + x_2 = x_3 + x_4 = x_5 + x_6 = x_7 + x_8 = x_9 + x_{10} = x_{11} + x_{12} = x_{13} + x_{14} = x_{15} + x_{16} \\ a_2 &= x_1 + x_3 = x_2 + x_4 = x_5 + x_7 = x_6 + x_8 = x_9 + x_{11} = x_{10} + x_{12} = x_{13} + x_{15} = x_{14} + x_{16} \\ a_3 &= x_1 + x_5 = x_2 + x_6 = x_3 + x_7 = x_4 + x_8 = x_9 + x_{13} = x_{10} + x_{14} = x_{11} + x_{15} = x_{12} + x_{16} \\ a_4 &= x_1 + x_9 = x_2 + x_{10} = x_3 + x_{11} = x_4 + x_{12} = x_5 + x_{13} = x_6 + x_{14} = x_7 + x_{15} = x_8 + x_{16} \end{aligned}$$

This allows use to define *Majority Logic Decoding*.

Suppose we receive the word $y_1y_2 \dots y_{15}y_{16}$ and that at most three errors occur during transmission. We evaluate the eight *check sums*

$$\{y_1 + y_2, y_3 + y_4, y_5 + y_6, y_7 + y_8, y_9 + y_{10}, y_{11} + y_{12}, y_{13} + y_{14}, y_{15} + y_{16}\}$$

If no errors occur during transmission then all eight check sums are the same (namely a_1). If at most three errors occur during transmission then at least five of these eight check sums will still equal a_1 (because y_i shows up in exactly one check sum for $1 \leq i \leq 16$). So we can decide the value of a_1 by a majority vote : it is the value that show up the most among those eight check sums.

Similarly, we can use a majority vote to recover the values of a_2, a_3 and a_4 .

It is very important to realize that we end up with the correct values of a_1, a_2, a_3 and a_4 if at most three errors occur during transmission.

Once we have the values of a_1, a_2, a_3 and a_4 , we can find the value of a_0 by another majority vote. Using (*) and the values we already have for a_1, a_2, a_3 and a_4 , we get sixteen check sums for a_0 . Another way of seeing this is as follows: after receiving the word $\mathbf{y} = y_1 \dots y_{16}$ and after finding the values of a_1, a_2, a_3 and a_4 , we find the word

$$z_1 \dots z_{16} = y_1 \dots y_{16} - [a_4 \ a_3 \ a_2 \ a_1 \ 0]G(1, 4)$$

Then a_0 equals the digit that shows up the most in $\{z_1, z_2, \dots, z_{15}, z_{16}\}$.

Again, we will end up with the correct value of a_0 if at most three errors occur during transmission.

If we really need to decode as a codeword, we decode as $[a_4 \ a_3 \ a_2 \ a_1 \ 0]G(1, 4)$ if $a_0 = 0$ and as $[a_4 \ a_3 \ a_2 \ a_1 \ 0]G(1, 4) + 1111111111111111$ if $a_0 = 1$.

If for some $0 \leq i \leq 4$ we can not decide the value of a_i by majority vote (because the number of check sums that equal 0 is the same as the number of the check sums that equal 1) then we know that at least four errors occurred during transmission.

Example : suppose we receive the word

$$1110110010111011$$

We calculate the check sums for a_1, a_2, a_3 and a_4 :

$$\begin{aligned}
a_1 & \quad \{0, 1, 0, 0, 1, 0, 1, 0\} \\
a_2 & \quad : \{0, 1, 1, 1, 0, 1, 0, 1\} \\
a_3 & \quad : \{0, 0, 1, 0, 0, 0, 0, 0\} \\
a_4 & \quad : \{0, 1, 0, 1, 0, 1, 1, 1\}
\end{aligned}$$

Hence we decide that $a_1 = 0$, $a_2 = 1$, $a_3 = 0$ and $a_4 = 1$.

To find a_0 , we calculate

$$11101100101111011 - [1 \ 0 \ 1 \ 0 \ 0] \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

We get

$$11101100101111011 - 0011001111001100 = 1101111101110111$$

So we decide that $a_0 = 1$.

Hence we decode as

$$0011001111001100 + 1111111111111111 = 1100110000110011$$

▷

4.4 Historical Note

First order Reed-Muller codes were used by NASA because of the very efficient decoding algorithm (namely *majority logic decoding*). In 1971, the spacecraft Mariner 9 was sent to Mars to study the Martian atmosphere and to map the planetary surface. Mariner 9 reached Mars on November 14, 1971 after 167 days of flight. It was the first man-made spacecraft to orbit another planet. Mariner 9 completed its mission in autumn 1972 and was turned off on October 27, 1972.

During its mission, Mariner 9 beamed back 7,329 black and white pictures of Mars of pixel size 832×700 . Each pixel contained one of 64 possible grayscale values (represented as a binary word of length 6). Digital data was transmitted using the first-order Reed-Muller code $R(1, 5)$ (which has 64 codewords, one for each grayscale). Note that $R(1, 5)$ can correct up to seven errors.

NASA assumed that the probability of a symbol error was 5%.

If only the original grayscale is transmitted (as a binary six-tuple) then the probability that the grayscale is received incorrectly is 26.49%. Due to power restrictions, NASA could transmit a grayscale using about thirty bits (five times the original number of bits). One naive method is to transmit the grayscale five times. So we use the repetition code $\{00000, 11111\}$ for each of the six bits (or digits) in the grayscale. The probability that a grayscale is decoded incorrectly now drops to 0.6929%. Using the Reed-Muller code $R(1, 5)$, the probability that the grayscale is decoded incorrectly drops even further down to 0.0139%. This implies that the probability that a decoded picture contains at most 116 incorrect pixels (less than 0.02% of the total picture) is 99.99%.

Chapter 5

Field Extensions

5.1 Extensions of the form $F(\alpha)$

Let F, K be fields such that $F \subset K$. Then we can view K as a vector space over F . We assume that $\dim_F K < +\infty$. Let $\alpha \in K$. Then there exists a monic polynomial $f(x)$ with coefficients in F (so $f(x) \in F[x]$) that is irreducible over F (so $f(x)$ can not be written as a product of polynomials with coefficients in F and of degree strictly less than the degree of $f(x)$) such that $f(\alpha) = 0$. If $n = \deg f(x)$ then

$$F(\alpha) = \{a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} : a_0, a_1, \dots, a_{n-1} \in F\}$$

Here the addition and multiplication are the usual polynomial operations combined with the fact that $f(\alpha) = 0$.

Example : Let $F = \mathbb{R}$ and $f(x) = x^2 + 1$. Define α such that $\alpha^2 + 1 = 0$. So $\alpha^2 = -1$. Then

$$\mathbb{R}(\alpha) = \{a + b\alpha : a, b \in \mathbb{R}\}$$

For the addition and multiplication we get that for all $a, b, c, d \in \mathbb{R}$

$$\begin{aligned}(a + b\alpha) + (c + d\alpha) &= (a + c) + (b + d)\alpha \\ (a + b\alpha)(c + d\alpha) &= ac + ad\alpha + bc\alpha + bd\alpha^2 = (ac - bd) + (ad + bc)\alpha\end{aligned}$$

So $\mathbb{R}(\alpha) = \mathbb{C}$. ▷

5.2 Finite Fields

If $F \subset K$ is a field extension then K is a vector space over F . Hence if K is a finite field in characteristic p then $|K|$ is a power of p . It turns out that for every prime power q , there exists a finite field K with $|K| = q$; moreover this field is unique up to isomorphism.

For each prime power q , we denote this (isomorphism type of) field by $GF(q)$: the *Galois field of size q* . If p is a prime then $GF(p) = \mathbb{Z}_p$, the integers modulo p .

Let p be a prime and $n \in \mathbb{N}$. How do we work with $GF(p^n)$?

We need a monic polynomial $f(x)$ of degree n with coefficients in $GF(p)$ that is irreducible over $GF(p)$. Define α by $f(\alpha) = 0$. Then

$$GF(p^n) = \{a_0 + a_1\alpha + \cdots + a_{n-1}\alpha^{n-1} : a_0, a_1, \dots, a_{n-1} \in GF(p)\}$$

as defined in Section 5.1.

Example : Consider $p = 2$ and $n = 3$. It is given that $x^3 + x + 1$ is irreducible over $GF(2)$ (we could actually verify this easily since a polynomial of degree three is irreducible over a field F if and only if it has no zeros in F). Define α by $\alpha^3 + \alpha + 1 = 0$. So $\alpha^3 = \alpha + 1$. Then

$$GF(8) = \{a_0 + a_1\alpha + a_2\alpha^2 : a_0, a_1, a_2 \in GF(2)\}$$

This allows us to perform calculations like

$$(1 + \alpha + \alpha^2) + (1 + \alpha) = \alpha^2$$

and

$$\begin{aligned} (1 + \alpha + \alpha^2)(1 + \alpha) &= 1 + \alpha + \alpha + \alpha^2 + \alpha^2 + \alpha^3 \\ &= 1 + \alpha^3 \\ &= 1 + \alpha + 1 \\ &= \alpha \end{aligned}$$

This ‘polynomial’ representation of $GF(8)$ makes addition very easy but multiplication quite cumbersome.

It turns out there is another way of representing a finite field. One can prove that $GF(q)^*$ (the multiplicative group of $GF(q)$) is a cyclic group of order $q - 1$. By choosing an appropriate irreducible polynomial $f(x)$, we get that α is a generator of $GF(q)^*$ if $f(\alpha) = 0$. If this is the case then

$$GF(q) = \{0, 1, \alpha, \alpha^2, \dots, \alpha^{q-2}\}$$

Such an element α is called a *primitive element of $GF(q)$* and such a polynomial $f(x)$ is called a *primitive polynomial*. This new representation makes multiplication very easy but it is not clear at all how to perform additions.

So we combine both representations in one table!

Example : It is given that $x^3 + x + 1$ is primitive over $GF(2)$. Define α by $\alpha^3 + \alpha + 1 = 0$. Then

$$GF(8) = \{a_0 + a_1\alpha + a_2\alpha^2 : a_0, a_1, a_2 \in GF(2)\} = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$$

To find out which power of α corresponds to which linear combination of 1 , α and α^2 , we perform the following calculations

$$\alpha^3 = \alpha + 1$$

$$\begin{aligned}\alpha^4 &= \alpha\alpha^3 = \alpha(\alpha + 1) = \alpha^2 + \alpha \\ \alpha^5 &= \alpha\alpha^4 = \alpha(\alpha^2 + \alpha) = \alpha^3 + \alpha^2 = \alpha + 1 + \alpha^2 = \alpha^2 + \alpha + 1 \\ \alpha^6 &= \alpha\alpha^5 = \alpha(\alpha^2 + \alpha + 1) = \alpha^3 + \alpha^2 + \alpha = \alpha + 1 + \alpha^2 + \alpha = \alpha^2 + 1\end{aligned}$$

We easily check that $\alpha^7 = 1$:

$$\alpha^7 = \alpha\alpha^6 = \alpha(\alpha^2 + 1) = \alpha^3 + \alpha = \alpha + 1 + \alpha = 1$$

We combine all this information in a table. The ‘binary’ column contains the coefficients of the polynomial expression : with $a_2\alpha^2 + a_1\alpha + a_0$ corresponds $a_2a_1a_0$.

binary	$GF(8)$	$a_2\alpha^2 + a_1\alpha + a_0$
000	0	0
001	1	1
010	α	α
100	α^2	α^2
011	α^3	$\alpha + 1$
110	α^4	$\alpha^2 + \alpha$
111	α^5	$\alpha^2 + \alpha + 1$
101	α^6	$\alpha^2 + 1$

This table allows us to switch between the ‘exponential’ notation (0 and the powers of α) and the ‘polynomial’ or ‘binary’ notation. Just keep in mind that $\alpha^7 = 1$.

- $\alpha^5 + \alpha^6 = 111 + 101 = 010 = \alpha$
- $\alpha^5\alpha^6 = \alpha^{11} = \alpha^7\alpha^4 = \alpha^4$
- $\frac{\alpha^4}{\alpha^6} = \alpha^{-2} = \alpha^{-2}\alpha^7 = \alpha^5$

5.3 Minimal Polynomial

Let $F \subset K$ be a field extension with $\dim_F K < +\infty$ and $\alpha \in K$. Then the set S of all monic polynomials with coefficients in F and α as a root is not empty. So

$$S = \{f(x) \in F[x] : f(\alpha) = 0\} \neq \emptyset$$

It turns out that S has exactly one element of smallest degree. This polynomial is called the *minimal polynomial of α over F* . We denote this minimal polynomial by $m_\alpha(x)$ where we assume that it is clear from the context what F is.

The following proposition contains two important properties of $m_\alpha(x)$ over F .

Proposition 5.1 *Let $F \subset K$ be a field extension with $\dim_F K < +\infty$ and $\alpha \in K$. Then the following holds:*

- $m_\alpha(x)$ is irreducible over F
- If $f(x) \in F[x]$ with $f(\alpha) = 0$ then $m_\alpha(x)$ divides $f(x)$.

5.4 Factoring $x^n - 1$ over $GF(q)$

Let q be a prime power and $n \in \mathbb{N}$ with $\gcd(n, q) = 1$.

In the next chapter, we will define cyclic codes, a very important class of linear codes. The construction of cyclic codes depends heavily on the factorization of $x^n - 1$ over $GF(q)$.

It turns out that there exists a field extension $GF(q^r)$ of $GF(q)$ in which $x^n - 1$ factors into linear factors. Indeed, let r be the order of q modulo n . So r is the smallest positive integer with $q^r \equiv 1 \pmod{n}$. Then n divides $q^r - 1$. But $GF(q^r)^*$ is a cyclic group of order $q^r - 1$. Since n divides $q^r - 1$, we get that $GF(q^r)^*$ contains a cyclic subgroup of order n . So let $\beta \in GF(q^r)$ be an element of order n . Such an element is also called a *primitive n -th root of unity*. Then $\{1, \beta, \beta^2, \dots, \beta^{n-1}\}$ are n different solutions of $x^n = 1$. Hence over $GF(q^r)$ we have that

$$x^n - 1 = (x - 1)(x - \beta)(x - \beta^2) \cdots (x - \beta^{n-1})$$

Example : Consider $x^9 - 1$ over $GF(2)$. We easily get that r (the order of 2 modulo 9) equals six : $2^6 \equiv 64 \equiv 1 \pmod{9}$. Note that $GF(64)^*$ is a cyclic group of order 63. Let $\beta \in GF(64)$ be a primitive 9-th root of unity. Then

$$x^9 - 1 = (x - 1)(x - \beta)(x - \beta^2) \cdots (x - \beta^8)$$

When working in $GF(64)$, we would start with $\alpha \in GF(64)$ of order 63 (namely a primitive element of $GF(64)$). Then we could choose $\beta = \alpha^7$. In fact, we have

$$\{1, \beta, \beta^2, \dots, \beta^8\} = \{1, \alpha^7, \alpha^{14}, \dots, \alpha^{56}\}$$

Note that we easily get that

$$x^9 - 1 = (x^3)^3 - 1 = (x^3 - 1)((x^3)^2 + x^3 + 1) = (x - 1)(x^2 + x + 1)(x^6 + x^3 + 1)$$

This leads to two questions :

- There exist $1 \leq i < j \leq 8$ such that

$$x^2 + x + 1 = (x - \beta^i)(x - \beta^j)$$

What are these values of i and j ?

- Is $x^6 + x^3 + 1$ irreducible over $GF(2)$? ▷

Going back to the general case, if

$$x^n - 1 = (x - 1)(x - \beta)(x - \beta^2) \cdots (x - \beta^{n-1})$$

which powers of β do we have to combine to get an irreducible factor over $GF(q)$? The answer is given by *cyclotomic cosets*.

Definition 5.2 Let q be a prime power, $n \in \mathbb{N}$ with $\gcd(n, q) = 1$ and $0 \leq s \leq n - 1$. The *cyclotomic coset of s , depending on q and n , (notation : C_s)* is the set

$$C_s = \{s, sq, sq^2, \dots\} \pmod n \quad \triangleright$$

Example : Let $q = 2$ and $n = 9$. For $0 \leq s \leq 8$, we denote by C_s the cyclotomic coset containing s . We easily get

$$\begin{aligned} C_0 &= \{0\} \\ C_1 = C_2 = C_4 = C_5 = C_7 = C_8 &= \{1, 2, 4, 8, 7, 5\} \\ C_3 = C_6 &= \{3, 6\} \end{aligned} \quad \triangleright$$

One can easily prove the following :

Proposition 5.3 *Let q be a prime power and $n \in \mathbb{N}$ with $\gcd(n, q) = 1$. Then the cyclotomic cosets depending on q and n partition $\{0, 1, \dots, n - 1\}$.*

The following theorem gives us a lot of information about the factorization of $x^n - 1$ over $GF(q)$.

Theorem 5.4 *Let q be a prime power, $n \in \mathbb{N}$ with $\gcd(n, q) = 1$ and β a primitive n -th root of unity. Then for $0 \leq s \leq n - 1$, we have that the minimal polynomial of β^s over $GF(q)$ is given by*

$$m_{\beta^s}(x) = \prod_{j \in C_s} (x - \beta^j)$$

Since minimal polynomials are irreducible, we get that $\prod_{j \in C_s} (x - \beta^j)$ is an irreducible factor over $GF(q)$ of $x^n - 1$.

Example : Let $q = 2$ and $n = 9$. We found earlier that there are three cyclotomic cosets :

$$\{0\}, \{1, 2, 4, 5, 7, 8\} \text{ and } \{3, 6\}$$

That means that the factorization of $x^9 - 1$ over $GF(2)$ contains three irreducible factors : one factor of degree one, one factor of degree two and one factor of degree six. We knew that

$$x^9 - 1 = (x - 1)(x^2 + x + 1)(x^6 + x^3 + 1)$$

So now we are assured that these factors are indeed irreducible. Moreover, if β is a primitive 9-th root of unity (in $GF(64)$) then it follows from the cyclotomic cosets that

$$\begin{aligned} x - 1 &= x - \beta^0 \\ x^2 + x + 1 &= (x - \beta^3)(x - \beta^6) \\ x^6 + x^3 + 1 &= (x - \beta)(x - \beta^2)(x - \beta^4)(x - \beta^5)(x - \beta^7)(x - \beta^8) \end{aligned}$$

We can check this using a table for $GF(64)$. If α is a primitive element in $GF(64)$ then

$$\beta \in \{\alpha^7, \alpha^{14}, \alpha^{28}, \alpha^{35}, \alpha^{49}, \alpha^{56}\}$$

and so

$$\{\beta^3, \beta^6\} = \{\alpha^{21}, \alpha^{42}\}$$

Hence

$$(x - \beta^3)(x - \beta^6) = (x - \alpha^{21})(x - \alpha^{42}) = x^2 - (\alpha^{21} + \alpha^{42}) + \alpha^{21}\alpha^{42} = x^2 - x + 1$$

since $\alpha^{21} + \alpha^{42} = 111011 + 111010 = 00001 = 1$ and $\alpha^{21}\alpha^{42} = \alpha^{63} = 1$.

What if we consider $n = 9$ and $q = 4$? Since $GF(2) \subset GF(4)$, it follows that a factorization of $x^9 - 1$ over $GF(2)$ is also a factorization over $GF(4)$ but the factors might not be irreducible anymore. So $x^6 + x^3 + 1$ is irreducible over $GF(2)$ but might not be irreducible over $GF(4)$. We easily get that the cyclotomic cosets depending on $n = 9$ and $q = 4$ are

$$\{0\}, \{1, 4, 7\}, \{2, 8, 5\}, \{3\} \text{ and } \{6\}$$

So the irreducible factors of $x^9 - 1$ over $GF(4)$ are

$$x - \beta^0, x - \beta^3, x - \beta^6, (x - \beta)(x - \beta^4)(x - \beta^7) \text{ and } (x - \beta^2)(x - \beta^5)(x - \beta^8)$$

where β is a primitive 9-th root of unity (in $GF(4)$).

Note that $GF(4)$ is a subfield of $GF(64)$. Indeed, if α is a primitive element in $GF(64)$ then

$$GF(4) = \{0, 1, \alpha^{21}, \alpha^{42}\}$$

Choosing $\beta = \alpha^7$, we get (after some calculations using the table for $GF(64)$) that

$$\begin{aligned} x - \beta^0 &= x - 1 \\ x - \beta^3 &= x - \alpha^{21} \\ x - \beta^6 &= x - \alpha^{42} \\ (x - \beta)(x - \beta^4)(x - \beta^7) &= x^3 - \alpha^{21} \\ (x - \beta^2)(x - \beta^5)(x - \beta^8) &= x^3 - \alpha^{42} \end{aligned}$$

So no matter how we work with $GF(4)$, say $GF(4) = \{0, 1, a, b\}$, we get that

$$x^9 - 1 = (x - 1)(x - a)(x - b)(x^3 - a)(x^3 - b) \quad \triangleright$$

Thus we can use cyclotomic coset to figure out how many irreducible factors there are in the factorization of $x^n - 1$ over $GF(q)$ and what the degrees are of these factors. In order to actually find these factors (as polynomials over $GF(q)$) we need a table for the field extension of $GF(q)$ that contains a primitive n -th root of unity.

5.5 The Ring $\mathbb{F}[x]$ Modulo $(x^n - 1)$

Let \mathbb{F} be a field and $n \in \mathbb{N}$. The ring $\mathbb{F}[x]$ modulo $(x^n - 1)$ is the quotient ring $\mathbb{F}[x]/(x^n - 1)$. So the elements of $\mathbb{F}[x]/(x^n - 1)$ are the cosets of $(x^n - 1)$ in $\mathbb{F}[x]$. Hence an element of $\mathbb{F}[x]/(x^n - 1)$ is not a polynomial but a set of polynomials, namely all the polynomials that have the same remainder when divided by $x^n - 1$. It turns out that every element of $\mathbb{F}[x]/(x^n - 1)$ contains exactly one polynomial of degree less than n . We can find this polynomial using long division. But it is easier to remember that $x^n \equiv 1 \pmod{(x^n - 1)}$.

Example : Consider the ring $GF(2)[x] \pmod{(x^3 - 1)}$.

Since $x^5 + x^4 + x^3 + x + 1 = (x^3 + 1)(x^2 + x + 1) + x^2$, we get that

$$x^5 + x^4 + x^3 + x + 1 \equiv x^2 \pmod{(x^3 - 1)}$$

But we did not have to use long division to get this result. Since $x^3 \equiv 1 \pmod{(x^3 - 1)}$, we have

$$x^5 + x^4 + x^3 + x + 1 \equiv x^2 + x + 1 + x + 1 \equiv x^2 \pmod{(x^3 - 1)} \quad \triangleright$$

Addition and multiplication in $\mathbb{F}[x]/(x^n - 1)$ are the usual polynomial operations with the additional 'rule' that $x^n = 1$.

Example : Consider the ring $GF(2)[x] \pmod{(x^3 - 1)}$. Then we have that

$$\begin{aligned} (x^2 + x) + (x + 1) &\equiv x^2 + 1 \pmod{(x^3 - 1)} \\ (x^2 + x)(x + 1) &\equiv x^3 + x^2 + x^2 + x \equiv x^3 + x \equiv 1 + x \pmod{(x^3 - 1)} \end{aligned} \quad \triangleright$$

Chapter 6

Cyclic Codes

6.1 Basic Definitions And Properties

Definition 6.1 Let \mathbb{F} be a field, $n \in \mathbb{N}$ and $\mathbf{y} = (y_0, y_1, \dots, y_{n-1}) \in \mathbb{F}^n$.

(a) The *cyclic shift of \mathbf{y} to the right* (notation : \mathbf{y}') is the word

$$\mathbf{y}' = (y_{n-1}, y_0, y_1, \dots, y_{n-2})$$

(b) The *word polynomial of \mathbf{y}* (notation : $\mathbf{y}(x)$) is the polynomial

$$\mathbf{y}(x) = y_0 + y_1x + y_2x^2 + \dots + y_{n-1}x^{n-1} \quad \triangleright$$

Remark : Let \mathbb{F} be a field, $\mathbf{v}, \mathbf{w} \in \mathbb{F}^n$ and $a, b \in \mathbb{F}$. Then $(a\mathbf{v} + b\mathbf{w})(x) = a\mathbf{v}(x) + b\mathbf{w}(x)$. \triangleright

Definition 6.2 Let C be an $[n, k]$ -code over \mathbb{F} . Then C is *cyclic* if the cyclic shift to the right of every codeword is again a codeword. \triangleright

Remark : Let C be a cyclic code. Then any cyclic shift of any codeword is a codeword. So if $(c_0, c_1, \dots, c_{n-1}) \in C$ then $(c_i, c_{i+1}, \dots, c_{n-1}, c_0, c_1, \dots, c_{i-1}) \in C$ for $i = 0, 1, \dots, n-1$. \triangleright

Example : The binary code $\{000, 110, 011, 101\}$ is cyclic. The set of codeword polynomials is $\{0, 1+x, x+x^2, 1+x^2\}$. \triangleright

The following proposition relates the word polynomial of a vector to the word polynomial of its shift.

Proposition 6.3 Let \mathbb{F} be a field, $n \in \mathbb{N}$ and $\mathbf{y} \in F^n$. Then $\mathbf{y}'(x) \equiv x \mathbf{y}(x) \pmod{(x^n - 1)}$.

Proof : Put $\mathbf{y} = (y_0, y_1, \dots, y_{n-1})$. Then

$$\begin{aligned} x \mathbf{y}(x) &= x(y_0 + y_1x + \dots + y_{n-2}x^{n-2} + y_{n-1}x^{n-1}) \\ &= y_0x + y_1x^2 + \dots + y_{n-2}x^{n-1} + y_{n-1}x^n \\ &= y_{n-1} + y_0x + \dots + y_{n-2}x^{n-1} + y_{n-1}(x^n - 1) \\ &= \mathbf{y}'(x) + y_{n-1}(x^n - 1) \end{aligned}$$

Hence $x \mathbf{y}(x) \equiv \mathbf{y}'(x) \pmod{(x^n - 1)}$. □

The next theorem shows the correspondence between cyclic codes and ideals in $\mathbb{F}[x]/(x^n - 1)$.

Theorem 6.4 Let \mathbb{F} be a field and $n \in \mathbb{N}$. Then the following holds:

- (a) Let C be a cyclic code of length n over \mathbb{F} . Then the set $I := \{\mathbf{c}(x) \pmod{(x^n - 1)} : \mathbf{c} \in C\}$ is an ideal of $\mathbb{F}[x]/(x^n - 1)$.
- (b) Let I be an ideal of $\mathbb{F}[x]/(x^n - 1)$. Then the set $C := \{\mathbf{y} \in \mathbb{F}^n : \mathbf{y}(x) \pmod{(x^n - 1)} \in I\}$ is a cyclic code of length n over \mathbb{F} .

Proof : (a) Let $\mathbf{v}, \mathbf{w} \in C$ and $a, b \in \mathbb{F}$. Then $a\mathbf{v}(x) + b\mathbf{w}(x) \pmod{(x^n - 1)} \in I$ since $a\mathbf{v}(x) + b\mathbf{w}(x) = (a\mathbf{v} + b\mathbf{w})(x)$ and $a\mathbf{v} + b\mathbf{w} \in C$.

Let $\mathbf{u} \in C$ and $f(x) \in \mathbb{F}[x]$. Put $f(x) = a_0 + a_1x + \dots + a_kx^k$. Then

$$\begin{aligned} f(x) \mathbf{u}(x) &\equiv (a_0 + a_1x + \dots + a_kx^k) \mathbf{u}(x) \pmod{(x^n - 1)} \\ &\equiv a_0\mathbf{u}(x) + a_1x \mathbf{u}(x) + \dots + a_kx^k \mathbf{u}(x) \pmod{(x^n - 1)} \end{aligned}$$

Note that $x \mathbf{u}(x) \equiv \mathbf{u}'(x) \pmod{(x^n - 1)}$. Since C is cyclic, we get that $\mathbf{u}' \in C$ and so $x \mathbf{u}(x) \pmod{(x^n - 1)} \in I$. Thus $x^2 \mathbf{u}(x) \equiv x \mathbf{u}'(x) \equiv \mathbf{u}''(x) \pmod{(x^n - 1)}$. Since $\mathbf{u}'' \in C$, we get that $x^2 \mathbf{u}(x) \pmod{(x^n - 1)} \in I$. Continuing this way, we see that $x^i \mathbf{u}(x) \pmod{(x^n - 1)} \in I$ for all $i \in \mathbb{N}$. It follows that $f(x) \mathbf{u}(x) \pmod{(x^n - 1)} \in I$. Hence I is an ideal of $\mathbb{F}[x]/(x^n - 1)$.

(b) Let $\mathbf{v}, \mathbf{w} \in C$ and $a, b \in \mathbb{F}$. Then $a\mathbf{v} + b\mathbf{w} \in C$ since $(a\mathbf{v} + b\mathbf{w})(x) = a\mathbf{v}(x) + b\mathbf{w}(x)$ and $a\mathbf{v}(x) + b\mathbf{w}(x) \pmod{(x^n - 1)} \in I$. So C is a linear code of length n over \mathbb{F} .

Let $\mathbf{u} = (u_0, u_1, \dots, u_{n-1}) \in C$. Then $\mathbf{u}' \in C$ since $\mathbf{u}'(x) \equiv x \mathbf{u}(x) \pmod{(x^n - 1)}$ and $x \mathbf{u}(x) \pmod{(x^n - 1)} \in I$. Hence C is cyclic. □

6.2 The Generator Polynomial

In the previous section, we saw that every cyclic code of length n over \mathbb{F} corresponds to an ideal of $\mathbb{F}[x]/(x^n - 1)$. Since $\mathbb{F}[x]/(x^n - 1)$ is a Principal Ideal Domain (that means that every ideal is generated by one element), we can construct cyclic codes if we know a polynomial that generates the corresponding ideal. Note that these ideals might have more than one generator. We will identify a specific generator called the *generator polynomial*.

Proposition 6.5 *Let C be a cyclic code of length n over \mathbb{F} and $g(x)$ a monic polynomial of smallest degree amongst all codeword polynomials. Then the following holds:*

- (a) $g(x)$ is unique.
- (b) $g(x)$ divides $\mathbf{c}(x)$ for all $\mathbf{c} \in C$.
- (c) $\{\mathbf{c}(x) \bmod (x^n - 1) : \mathbf{c} \in C\} = \langle g(x) \bmod (x^n - 1) \rangle$.
- (d) $g(x)$ divides $x^n - 1$.

Proof : Note that $(a\mathbf{c})(x) = a\mathbf{c}(x)$ for all $\mathbf{c} \in C$ and all $a \in \mathbb{F}$. Hence there are monic codeword polynomials.

(a) Suppose that $h(x)$ is a monic codeword polynomial of smallest degree. Let $\mathbf{v}, \mathbf{w} \in C$ with $g(x) = \mathbf{v}(x)$ and $h(x) = \mathbf{w}(x)$. Then $g(x) - h(x) = \mathbf{v}(x) - \mathbf{w}(x) = (\mathbf{v} - \mathbf{w})(x)$. But $\mathbf{v} - \mathbf{w} \in C$. If $g(x) \neq h(x)$ then there exists a non-zero codeword polynomial (and hence also a monic codeword polynomial) of degree smaller than $\deg(g(x))$, a contradiction to the choice of $g(x)$. Hence $g(x) = h(x)$ and so $g(x)$ is unique.

(b)(c) Put $I = \{\mathbf{c}(x) \bmod (x^n - 1) : \mathbf{c} \in C\}$. Then I is an ideal of $\mathbb{F}[x]/(x^n - 1)$ by Theorem 6.4(a). Since $g(x) = \mathbf{u}(x)$ for some $\mathbf{u} \in C$, we get that $\langle g(x) \bmod (x^n - 1) \rangle \subseteq I$. Let $\mathbf{c} \in C$. Let $q(x)$ (resp. $r(x)$) be the quotient (resp. remainder) of $\mathbf{c}(x)$ divided by $g(x)$. So $\mathbf{c}(x) = g(x)q(x) + r(x)$ and either $r(x) = 0$ or $\deg(r(x)) < \deg(g(x))$. Note that

$$r(x) \equiv \mathbf{c}(x) - g(x)q(x) \pmod{x^n - 1}$$

So $r(x) \bmod (x^n - 1) \in I$. Thus $r(x) \equiv \mathbf{v}(x) \bmod (x^n - 1)$ for some $\mathbf{v} \in C$. This is only possible if $r(x) = \mathbf{v}(x)$ since $\deg(r(x)), \deg(\mathbf{v}(x)) < n$ or $r(x), \mathbf{v}(x) = 0$. If $r(x) \neq 0$ then there exists a non-zero codeword polynomial (and hence also a monic codeword polynomial) of degree smaller than $\deg(g(x))$, a contradiction to the choice of $g(x)$. Hence $r(x) = 0$. So $\mathbf{c}(x) = g(x)q(x)$, which proves (b). Thus $\mathbf{c}(x) \equiv g(x)q(x) \pmod{x^n - 1}$ and $\mathbf{c}(x) \bmod (x^n - 1) \in \langle g(x) \bmod (x^n - 1) \rangle$. So $I \subseteq \langle g(x) \bmod (x^n - 1) \rangle$. Hence $I = \langle g(x) \bmod (x^n - 1) \rangle$, which proves (c).

(d) Let $q(x)$ (resp. $r(x)$) be the quotient (resp. remainder) of $x^n - 1$ divided by $g(x)$. So $x^n - 1 = g(x)q(x) + r(x)$ and either $r(x) = 0$ or $\deg(r(x)) < \deg(g(x))$. Hence

$$r(x) \equiv x^n - 1 - g(x)q(x) \equiv g(x)(-q(x)) \pmod{x^n - 1}$$

So it follows from (c) that $r(x) \equiv \mathbf{c}(x) \bmod (x^n - 1)$ for some $\mathbf{c} \in C$. This is only possible if $r(x) = \mathbf{c}(x)$. If $r(x) \neq 0$ then there exists a non-zero codeword polynomial (and hence also a monic codeword polynomial) of degree smaller than $\deg(g(x))$, a contradiction to the choice of $g(x)$. Hence $r(x) = 0$. So $x^n - 1 = g(x)q(x)$ and $g(x)$ divides $x^n - 1$. \square

Example : Consider the binary cyclic code $\{000, 110, 011, 101\}$. Then the codeword polynomials are $\{0, 1 + x, x + x^2, 1 + x^2\}$. The generator polynomial is $1 + x$ and every codeword polynomial is a multiple of $1 + x$. Indeed, $x + x^2 = x(1 + x)$ and $1 + x^2 = (1 + x)(1 + x)$. Finally, $\{0, 1 + x, x + x^2, 1 + x^2\} \bmod (x^3 - 1) = \langle 1 + x \bmod (x^3 - 1) \rangle$.

Note that the ideal associated with this code has other generators. Since $x(1+x^2) \equiv x+x^3 \equiv x+1 \pmod{(x^3-1)}$ and $x^2(1+x^2) \equiv x^2+x^4 \equiv x^2+x \pmod{(x^3-1)}$, we get that $\{0, 1+x, x+x^2, 1+x^2\} \pmod{(x^3-1)} = \langle 1+x^2 \pmod{(x^3-1)} \rangle$. \triangleright

A cyclic code is completely determined by its generator polynomial. Every monic divisor of $x^n - 1$ determines a cyclic code. Hence the number of cyclic codes of length n is the number of monic divisors of $x^n - 1$.

Example : We want to find all the binary cyclic codes of length 7.

We start by calculating the cyclotomic cosets depending on $n = 7$ and $q = 2$. We easily get

$$\{0\}, \{1, 2, 4\} \text{ and } \{3, 6, 5\}$$

We use the table for $GF(8)$ to actually find the irreducible factors. Let $\beta \in GF(8)$ be an element of order 7. So we can choose $\beta = \alpha$ where α is the primitive element from the table. Then the irreducible factors over $GF(2)$ are

$$x - \beta^0, (x - \beta^1)(x - \beta^2)(x - \beta^4) \text{ and } (x - \beta^3)(x - \beta^5)(x - \beta^6)$$

Using the table for $GF(8)$ if needed, we get

$$x - \beta^0 = x + 1$$

$$\begin{aligned} (x - \beta^1)(x - \beta^2)(x - \beta^4) &= x^3 + (\alpha + \alpha^2 + \alpha^4)x^2 + (\alpha^3 + \alpha^6 + \alpha^5)x + \alpha^7 \\ &= x^3 + (\alpha + \alpha^2 + \alpha^4)x^2 + (\alpha^3 + \alpha^6 + \alpha^5)x + 1 \\ &= x^3 + x + 1 \end{aligned}$$

$$\begin{aligned} (x - \beta^3)(x - \beta^5)(x - \beta^6) &= x^3 + (\alpha^3 + \alpha^5 + \alpha^6)x^2 + (\alpha^8 + \alpha^{11} + \alpha^9)x + \alpha^{14} \\ &= x^3 + (\alpha^3 + \alpha^5 + \alpha^6)x^2 + (\alpha + \alpha^4 + \alpha^2)x + 1 \\ &= x^3 + x^2 + 1 \end{aligned}$$

since

$$\alpha + \alpha^2 + \alpha^4 = 010 + 100 + 110 = 000 = 0$$

$$\alpha^3 + \alpha^6 + \alpha^5 = 011 + 101 + 111 = 001 = 1$$

Hence the factorization of $x^7 - 1$ into irreducible factors over $GF(2)$ is

$$x^7 - 1 = (x + 1)(x^3 + x^2 + 1)(x^3 + x + 1)$$

So there are three different irreducible factors. Hence $x^7 - 1$ has $2^3 = 8$ different (monic) divisors. So there are eight binary cyclic codes of length 7. The generator polynomials for these eight codes are

Code	Generator Polynomial
C_0	1
C_1	$1 + x$
C_2	$1 + x + x^3$
C_3	$1 + x^2 + x^3$
C_4	$(1 + x)(1 + x + x^3)$
C_5	$(1 + x)(1 + x^2 + x^3)$
C_6	$(1 + x + x^3)(1 + x^2 + x^3)$
C_7	$(1 + x)(1 + x + x^3)(1 + x^2 + x^3)$

Note that $C_0 = \{0, 1\}^7$ (namely the set of all binary words of length 7) while $C_7 = \{0000000\}$ (note that the generator polynomial for C_7 is $1 + x^7$). \triangleright

6.3 Generator Matrix for a Cyclic Code

Using the generator polynomial of a cyclic code, we can easily find the dimension and a generator matrix of the code.

Theorem 6.6 *Let C be a cyclic code of length n over \mathbb{F} and $g(x)$ the generator polynomial of C . Then the following holds:*

- (a) *The dimension of C is $n - \deg(g(x))$.*
- (b) *If $g(x) = a_0 + a_1x + \dots + a_tx^t$ then following matrix (with n columns)*

$$\begin{bmatrix} a_0 & a_1 & a_2 & \cdots & a_t & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & a_0 & a_1 & \cdots & a_{t-1} & a_t & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & a_0 & \cdots & a_{t-2} & a_{t-1} & a_t & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_t & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & a_{t-1} & a_t \end{bmatrix}$$

is a generator matrix for C .

Proof : (a)(b) Note that the given matrix G is an $(n-t) \times n$ -matrix. So (a) follows from (b). For $1 \leq i \leq n-t$, let \mathbf{c}_i be the word given by the i -th row of G . So $\mathbf{c}_1 = (a_0, a_1, \dots, a_{t-1}, a_t, 0, \dots, 0)$. Since $g(x)$ is the generator polynomial of C , we have that $\mathbf{c}_1 \in C$. Since C is cyclic, we get that the cyclic shift of \mathbf{c}_1 to the right is also a codeword. So $\mathbf{c}_2 \in C$. Continuing this way, we see that $\mathbf{c}_i \in C$ for all $1 \leq i \leq n-t$. We will show that $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{n-t}\}$ is a basis for C . Since $a_t = 1$, we get that

$$\begin{bmatrix} 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ a_{t-1} & 1 & 0 & 0 & \cdots & 0 & 0 \\ a_{t-2} & a_{t-1} & 1 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \cdots & \cdots & \cdots & \cdots & \cdots & 1 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & a_{t-1} & 1 \end{bmatrix}$$

is a non-singular submatrix of G . So the rank of G is $n - t$. Hence the rows of G are linearly independent. So $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{n-t}\}$ is linearly independent.

Let $\mathbf{0} \neq \mathbf{c} \in C$. It follows from proposition 6.5(b) that $\mathbf{c}(x) = g(x)f(x)$ for some polynomial $f(x) \in \mathbb{F}[x]$. So $\deg(f(x)) = \deg(\mathbf{c}(x)) - \deg(g(x)) \leq n - 1 - t$. Put $f(x) = \lambda_0 + \lambda_1x + \dots + \lambda_{n-t-1}x^{n-t-1}$. Then

$$\mathbf{c}(x) = f(x)g(x) = \lambda_0g(x) + \lambda_1(xg(x)) + \dots + \lambda_{n-t-1}(x^{n-t-1}g(x))$$

But $x^i g(x) = \mathbf{c}_{i+1}(x)$ for $0 \leq i \leq n - t - 1$. Hence

$$\mathbf{c} = \lambda_0\mathbf{c}_1 + \lambda_1\mathbf{c}_2 + \dots + \lambda_{n-t-1}\mathbf{c}_{n-t}$$

So $\text{span}\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{n-t}\} = C$.

Hence $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{n-t}\}$ is a basis for C . Thus G is a generator matrix for C . \square

Example : Consider the binary cyclic code C of length 7 with generator polynomial $(1+x)(1+x+x^3) = 1+x^2+x^3+x^4$. Then a generator matrix for C is

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

Note that

$$C = \{0000000, 1011100, 0101110, 0010111, 1001011, 1100101, 1110010, 0111001\} \quad \triangleright$$

6.4 Parity Check Matrix for a Cyclic Code

Using the definition of cyclic codes, one can prove that the dual of a cyclic code is cyclic. So in order to find a parity check matrix for a cyclic code, we need to find the generator polynomial of the dual code.

Definition 6.7 Let $f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-2}x^{k-2} + a_{k-1}x^{k-1} + a_kx^k$ be a polynomial of degree k . Then the *reciprocal of f* (notation : $f^*(x)$) is the polynomial

$$f^*(x) = a_k + a_{k-1}x + a_{k-2}x^2 + \dots + a_2x^{k-2} + a_1x^{k-1} + a_0x^k \quad \triangleright$$

Example : If $f(x) = 1 + x + x^4 + x^6$ then $f^*(x) = 1 + x^2 + x^5 + x^6$. If $g(x) = x^2 + x^3$ then $g^*(x) = 1 + x$. \triangleright

Remark : If $f(x)$ is a polynomial of degree k then $f^*(x) = x^k f\left(\frac{1}{x}\right)$. Indeed, let $f(x) = a_0 + a_1x + a_2x^2 + \dots + a_{k-2}x^{k-2} + a_{k-1}x^{k-1} + a_kx^k$, then

$$\begin{aligned} x^k f\left(\frac{1}{x}\right) &= x^k \left(a_0 + \frac{a_1}{x} + \frac{a_2}{x^2} + \dots + \frac{a_{k-2}}{x^{k-2}} + \frac{a_{k-1}}{x^{k-1}} + \frac{a_k}{x^k} \right) \\ &= a_0x^k + a_1x^{k-1} + a_2x^{k-2} + \dots + a_{k-2}x^2 + a_{k-1}x + a_k \\ &= f^*(x) \end{aligned} \quad \triangleright$$

Lemma 6.8 Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$. Then \mathbf{a} is perpendicular to \mathbf{b} and all the cyclic shifts of \mathbf{b} if and only if $\mathbf{a}(x)\mathbf{b}^*(x) \equiv 0 \pmod{(x^n - 1)}$.

Proof : Put $\mathbf{a} = (a_0, a_1, \dots, a_{n-1})$ and $\mathbf{b} = (b_0, b_1, \dots, b_{n-1})$. Put

$$f(x) = b_{n-1} + b_{n-2}x + b_{n-3}x^2 + \dots + b_2x^{n-3} + b_1x^{n-2} + b_0x^{n-1}$$

Note that $f(x) = x^t\mathbf{b}^*(x)$ where $t = n - 1 - \deg(\mathbf{b}(x))$. Since $\gcd(x^t, x^n - 1) = 1$ we get that $\mathbf{a}(x)\mathbf{b}^*(x) \equiv 0 \pmod{(x^n - 1)} \Leftrightarrow \mathbf{a}(x)x^t\mathbf{b}^*(x) \equiv 0 \pmod{(x^n - 1)} \Leftrightarrow \mathbf{a}(x)f(x) \equiv 0 \pmod{(x^n - 1)}$

We have that

$$\mathbf{a}(x)f(x) \equiv (a_0 + a_1x + \dots + a_{n-2}x^{n-2} + a_{n-1}x^{n-1})(b_{n-1} + b_{n-2}x + \dots + b_1x^{n-2} + b_0x^{n-1}) \pmod{(x^n - 1)}$$

Let $0 \leq i \leq n - 1$. What is the coefficient of x^i in $\mathbf{a}(x)f(x) \pmod{(x^n - 1)}$? For $0 \leq j, k \leq n - 1$, we get that the coefficient of x^j in $\mathbf{a}(x)$ is a_j and the coefficient of x^k in $f(x)$ is b_{n-1-k} ; moreover $x^jx^k \equiv x^i \pmod{(x^n - 1)} \Leftrightarrow j + k \equiv i \pmod{n}$. Hence

$$\mathbf{a}(x)f(x) \equiv \sum_{i=0}^{n-1} \left(\sum_{j=0}^{n-1} a_j b_{j-i-1 \pmod{n}} \right) x^i \pmod{(x^n - 1)}$$

So

$$\mathbf{a}(x)f(x) \equiv 0 \pmod{(x^n - 1)} \Leftrightarrow \sum_{j=0}^{n-1} a_j b_{j-i-1 \pmod{n}} = 0 \quad \text{for } 0 \leq i \leq n - 1$$

For $0 \leq i \leq n - 1$, we get that

$$\begin{aligned} \sum_{j=0}^{n-1} a_j b_{j-i-1 \pmod{n}} &= a_0 b_{n-i-1} + a_1 b_{n-i} + \dots + a_{i-1} b_{n-2} + a_i b_{n-1} \\ &\quad + a_{i+1} b_0 + a_{i+2} b_1 + \dots + a_{n-2} b_{n-i-3} + a_{n-1} b_{n-i-2} \\ &= (a_0, a_1, \dots, a_{n-2}, a_{n-1}) \cdot (b_{n-i-1}, b_{n-i}, \dots, b_{n-2}, b_{n-1}, b_0, b_1, \dots, b_{n-i-3}, b_{n-i-2}) \\ &= \mathbf{a} \cdot \mathbf{b}^{(i+1)} \end{aligned}$$

where $\mathbf{b}^{(i+1)}$ is the the word we obtain by shifting \mathbf{b} to the right $(i + 1)$ times.

So $\mathbf{a}(x)\mathbf{b}^*(x) \equiv 0 \pmod{(x^n - 1)}$ if and only if \mathbf{a} is orthogonal to \mathbf{b} and all its cyclic shifts. \square

Lemma 6.9 Let $\mathbf{a}, \mathbf{b} \in \mathbb{F}^n$. If \mathbf{a} is orthogonal to \mathbf{b} and all its cyclic shifts then any cyclic shift of \mathbf{a} is orthogonal to any cyclic shift of \mathbf{b} .

Proof : For $\mathbf{c} \in \mathbb{F}^n$ and $k \in \mathbb{N}$, we denote by $\mathbf{c}^{(k)}$ the k -th cyclic shift to the right of \mathbf{c} . It's easy to verify that $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}' \cdot \mathbf{y}'$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$. So $\mathbf{x} \cdot \mathbf{y} = \mathbf{x}^{(k)} \cdot \mathbf{y}^k$ for all $\mathbf{x}, \mathbf{y} \in \mathbb{F}^n$ and all $k \in \mathbb{N}$. Let $0 \leq i, j \leq n - 1$. Then

$$\mathbf{a}^{(i)} \cdot \mathbf{b}^{(j)} = (\mathbf{a}^{(i)})^{(n-i)} \cdot (\mathbf{b}^{(j)})^{(n-i)} = \mathbf{a} \cdot \mathbf{b}^{n-i+j} = 0$$

So any cyclic shift of \mathbf{a} is orthogonal to any cyclic shift of \mathbf{b} . \square

The next definition is related to the generator polynomial of the dual of a cyclic code.

Definition 6.10 Let C be a cyclic code with generator polynomial $g(x)$. Since $g(x)$ divides $x^n - 1$, we have that $x^n - 1 = g(x)h(x)$ for some polynomial $h(x)$. We call $h(x)$ the *check polynomial* of C . \triangleright

Example : Let C be binary cyclic code of length 7 with generator polynomial $g(x) = 1 + x^2 + x^3$. Since

$$x^7 - 1 = (x + 1)(x^3 + x + 1)(x^3 + x^2 + 1)$$

we get that the check polynomial of C is

$$h(x) = (x + 1)(x^3 + x + 1) = x^4 + x^3 + x^2 + 1 \quad \triangleright$$

Theorem 6.11 Let C be a cyclic code of length n over \mathbb{F} with check polynomial $h(x)$. Then the following holds:

(a) C^\perp is a cyclic code with generator polynomial $\frac{1}{h(0)} h^*(x)$.

(b) If $h(x) = b_0 + b_1x + \cdots + b_kx^k$ then the following matrix (with n columns)

$$\begin{bmatrix} b_k & b_{k-1} & b_{k-2} & \cdots & b_0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & b_k & b_{k-1} & \cdots & b_1 & b_0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & b_k & \cdots & b_2 & b_1 & b_0 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & b_0 & 0 \\ 0 & 0 & 0 & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & b_1 & b_0 \end{bmatrix}$$

is a parity check matrix for C .

Proof : (a)(b) Let $g(x)$ be the generator polynomial of C . Since $x^n - 1 = g(x)h(x)$, it follows from Theorem 6.6(a) that $\dim C = n - \deg(g(x)) = \deg(h(x)) = k$. Let $\mathbf{a} \in C$ such that $\mathbf{a}(x) = g(x)$. Then it follows from Theorem 6.6(b) that $\{\mathbf{a}, \mathbf{a}', \mathbf{a}'', \dots, \mathbf{a}^{(k-1)}\}$ is a basis for C . Put $h(x) = b_0 + b_1x + \cdots + b_kx^k$. Then

$$f(x) := \frac{1}{h(0)} h^*(x) = \frac{1}{b_0} (b_k + b_{k-1}x + \cdots + b_1x^{k-1} + b_0x^k) = \frac{b_k}{b_0} + \frac{b_k}{b_0}x + \cdots + \frac{b_1}{b_0}x^{k-1} + x^k$$

So $f(x)$ is monic. Note that $x^n - 1 = g(x)h(x)$. Substituting x by $\frac{1}{x}$ and multiplying by x^n , we find that

$$1 - x^n = x^n \left(\frac{1}{x^n} - 1 \right) = x^n g \left(\frac{1}{x} \right) h \left(\frac{1}{x} \right) = \left[x^{n-k} g \left(\frac{1}{x} \right) \right] \left[x^k h \left(\frac{1}{x} \right) \right] = g^*(x) h^*(x)$$

So $h^*(x)$ (and thus also $f(x)$) is a divisor of $x^n - 1$.

Hence $f(x)$ is the generator polynomial of a cyclic code C^* . We claim that $C^* = C^\perp$. It follows from Theorem 6.6(a) that $\dim C^* = n - \deg(f(x)) = n - k$. Let $\mathbf{b} \in C^*$ with $\mathbf{b}(x) = h^*(x) = b_0f(x)$. Then $\{\mathbf{b}, \mathbf{b}', \mathbf{b}'', \dots, \mathbf{b}^{n-k-1}\}$ is a basis for C^* by Theorem 6.6(b). Note that

$$\mathbf{a}(x)\mathbf{b}^*(x) \equiv g(x)h^{**}(x) \equiv g(x)h(x) \equiv x^n - 1 \equiv 0 \pmod{x^n - 1}$$

So by Lemmas 6.8 and 6.9, we have that any shift of \mathbf{a} is orthogonal to any shift of \mathbf{b} . So $\mathbf{a}^{(i)}$ is orthogonal to $\mathbf{b}^{(j)}$ for $0 \leq i \leq k-1$ and $0 \leq j \leq n-k-1$.

But $\{\mathbf{a}, \mathbf{a}', \mathbf{a}'', \dots, \mathbf{a}^{(k-1)}\}$ is a basis for C and $\{\mathbf{b}, \mathbf{b}', \mathbf{b}'', \dots, \mathbf{b}^{(n-k-1)}\}$ is a basis for C^* . So any vector in C^* is orthogonal to any vector in C . Hence $C^* \subseteq C^\perp$. Since $\dim C^* = n-k = n - \dim C = \dim C^\perp$, we must have that $C^\perp = C^*$. \square

Example : Let C be the binary cyclic code of length 7 with generator polynomial $x^3 + x^2 + 1$.

Then

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

is a generator matrix for C .

Moreover, since

$$x^7 - 1 = (x+1)(x^3 + x + 1)(x^3 + x^2 + 1) = (x^3 + x^2 + 1)(x^4 + x^3 + x^2 + 1) = (x^3 + x^2 + 1)h(x)$$

we get that $h^*(x) = x^4 + x^2 + x + 1$ is the generator polynomial for C^\perp . Hence

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

is a parity check matrix for C .

\triangleright

Chapter 7

Bose-Chaudhuri-Hocquenghem Codes

In this chapter, we define a special class of cyclic codes. These codes are quite practical since efficient decoding algorithms are known.

7.1 Definitions

Definition 7.1 Let $GF(q)$ be a finite field, $n \in \mathbb{N}$ with $\gcd(n, q) = 1$ and $2 \leq \delta \leq n$. A *Bose-Chaudhuri-Hocquenghem code* (short : *BCH-code*) of length n over $GF(q)$ of designed distance δ is a cyclic code of length n over $GF(q)$ with generator polynomial

$$g(x) = \text{lcm}(m_{\beta^b}(x), m_{\beta^{b+1}}(x), \dots, m_{\beta^{b+\delta-2}}(x))$$

where β is an element of order n in a field extension of $GF(q)$ and b is a positive integer. \triangleright

Example : We will construct a binary BCH-code of length 9 and designed distance 3. So the generator polynomial $g_b(x)$ is of the form

$$g_b(x) = \text{lcm}(m_{\beta^b}(x), m_{\beta^{b+1}}(x))$$

where β is an element of order 9 in some field extension of $GF(2)$ (namely $GF(64)$) and $m_{\beta^j}(x)$ is the minimal polynomial over $GF(2)$ of β^j .

Recall that the cyclotomic cosets depending on $n = 9$ and $q = 2$ are

$$\{0\} \quad , \quad \{1, 2, 4, 5, 7, 8\} \quad \text{and} \quad \{3, 6\}$$

The corresponding irreducible factors of $x^9 - 1$ are

$$x + 1 \quad , \quad x^6 + x^3 + 1 \quad \text{and} \quad x^2 + x + 1$$

So

$$\begin{aligned} m_{\beta^0}(x) &= x + 1 \\ m_{\beta^1}(x) &= m_{\beta^2}(x) = m_{\beta^4}(x) = m_{\beta^5}(x) = m_{\beta^7}(x) = m_{\beta^8}(x) = x^6 + x^3 + 1 \\ m_{\beta^3}(x) &= m_{\beta^6}(x) = x^2 + x + 1 \end{aligned}$$

The BCH-code depends heavily on the choice of b :

b	generator polynomial $g_b(x)$		
0	$\text{lcm}(m_{\beta^0}(x), m_{\beta^1}(x))$	$= \text{lcm}(x + 1, x^6 + x^3 + 1)$	$= (x + 1)(x^6 + x^3 + 1)$
1	$\text{lcm}(m_{\beta^1}(x), m_{\beta^2}(x))$	$= \text{lcm}(x^6 + x^3 + 1, x^6 + x^3 + 1)$	$= x^6 + x^3 + 1$
2	$\text{lcm}(m_{\beta^2}(x), m_{\beta^3}(x))$	$= \text{lcm}(x^6 + x^3 + 1, x^2 + x + 1)$	$= (x^6 + x^3 + 1)(x^2 + x + 1)$
3	$\text{lcm}(m_{\beta^3}(x), m_{\beta^4}(x))$	$= \text{lcm}(x^2 + x + 1, x^6 + x^3 + 1)$	$= (x^2 + x + 1)(x^6 + x^3 + 1)$
4	$\text{lcm}(m_{\beta^4}(x), m_{\beta^5}(x))$	$= \text{lcm}(x^6 + x^3 + 1, x^6 + x^3 + 1)$	$= x^6 + x^3 + 1$
5	$\text{lcm}(m_{\beta^5}(x), m_{\beta^6}(x))$	$= \text{lcm}(x^6 + x^3 + 1, x^2 + x + 1)$	$= (x^6 + x^3 + 1)(x^2 + x + 1)$
6	$\text{lcm}(m_{\beta^6}(x), m_{\beta^7}(x))$	$= \text{lcm}(x^2 + x + 1, x^6 + x^3 + 1)$	$= (x^2 + x + 1)(x^6 + x^3 + 1)$
7	$\text{lcm}(m_{\beta^7}(x), m_{\beta^8}(x))$	$= \text{lcm}(x^6 + x^3 + 1, x^6 + x^3 + 1)$	$= x^6 + x^3 + 1$
8	$\text{lcm}(m_{\beta^8}(x), m_{\beta^9}(x))$	$= \text{lcm}(x^6 + x^3 + 1, x + 1)$	$= (x^6 + x^3 + 1)(x + 1)$

So using cyclotomic cosets, we can easily find the degree of the generator polynomial of a BCH-code and hence also the dimension of the code. \triangleright

In general, finding the minimum distance of a BCH-code is quite hard. We will prove in the next section that the designed distance δ is only a lower bound for the true minimum distance.

Definition 7.2 A Reed-Solomon Code over $GF(q)$ of designed distance δ is a BCH-code over $GF(q)$ of length $n = q - 1$ and designed distance δ . \triangleright

Remark : Let C be a Reed-Solomon code over $GF(q)$ of designed distance δ . Let β be an element of order $n = q - 1$. Then $\beta \in GF(q)$ (in fact : β is a primitive element of $GF(q)$); so we can choose $\beta = \alpha$ if we have a table for $GF(q)$). Hence the minimal polynomial $m_{\beta^j}(x)$ of β^j over $GF(q)$ is $m_{\beta^j}(x) = x - \beta^j$. So the generator polynomial $g(x)$ of C is

$$g(x) = \text{lcm}(m_{\beta^b}(x), m_{\beta^{b+1}}(x), \dots, m_{\beta^{b+\delta-2}}(x)) = (x - \beta^b)(x - \beta^{b+1}) \dots (x - \beta^{b+\delta-2})$$

Hence $\deg(g(x)) = \delta - 1$ and $\dim C = n - \deg(g(x)) = (q - 1) - (\delta - 1) = q - \delta$. \triangleright

Example : We will construct a Reed-Solomon code C over $GF(16)$ of designed distance 5. We choose $\bar{b} = 1$. Then the generator polynomial $g(x)$ of C is

$$g(x) = (x - \alpha)(x - \alpha^2)(x - \alpha^3)(x - \alpha^4)$$

where α is a primitive element of $GF(16)$. So C is a cyclic code of length 15 over $GF(16)$ of dimension $15 - 4 = 11$.

If we really want to know this generator polynomial, we have to use a table for $GF(16)$. After some calculation, we find

$$g(x) = x^4 + \alpha^5 x^3 + \alpha^4 x^2 + \alpha^{12} x + \alpha^{10} \quad \triangleright$$

Unlike general BCH-codes, the designed minimum distance δ of a Reed-Solomon code is the actual minimum distance.

7.2 Vandermonde Matrices

In this section, we introduce the concept of a Vandermonde matrix and prove a closed formula for its determinant. We will use these results in the next sections to get more information about the minimum distance of a BCH-code and to prove a decoding algorithm for BCH-codes.

Definition 7.3 Let x_1, x_2, \dots, x_n be n numbers. Then the *Vandermonde matrix determined by x_1, x_2, \dots, x_n* is the square $n \times n$ -matrix

$$V(x_1, x_2, \dots, x_n) := \begin{bmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{bmatrix}$$

Sometimes the transposed of this matrix is called a Vandermonde matrix as well. ▷

It turns out that there is a closed formula for the determinant of a Vandermonde matrix.

Proposition 7.4 Let x_1, x_2, \dots, x_n be n numbers. Then

$$\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix} = \prod_{1 \leq i < j \leq n} (x_j - x_i)$$

Proof : We prove this formula by induction on n .

For $n = 2$, we get that

$$\begin{vmatrix} 1 & 1 \\ x_1 & x_2 \end{vmatrix} = x_2 - x_1 = \prod_{1 \leq i < j \leq 2} (x_j - x_i)$$

So assume that $\det(V(a_1, a_2, \dots, a_k)) = \prod_{1 \leq i < j \leq k} (a_j - a_i)$ for all $k = 2, 3, \dots, n-1$ where $n \geq 3$ and all numbers a_1, a_2, \dots, a_k .

Consider the determinant

$$V := \begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix}$$

We perform the following elementary row operations:

$$R_i \leftrightarrow R_i - x_1 R_{i-1} \quad \text{for } i = n, n-1, \dots, 2$$

Then we get

$$V = \begin{vmatrix} 1 & 1 & 1 & \cdots & 1 \\ 0 & x_2 - x_1 & x_3 - x_1 & \cdots & x_n - x_1 \\ 0 & x_2^2 - x_1x_2 & x_3^2 - x_1x_3 & \cdots & x_n^2 - x_1x_n \\ 0 & x_2^3 - x_1x_2^2 & x_3^3 - x_1x_3^2 & \cdots & x_n^3 - x_1x_n^2 \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & x_2^{n-2} - x_1x_2^{n-3} & x_3^{n-2} - x_1x_3^{n-3} & \cdots & x_n^{n-2} - x_1x_n^{n-3} \\ 0 & x_2^{n-1} - x_1x_2^{n-2} & x_3^{n-1} - x_1x_3^{n-2} & \cdots & x_n^{n-1} - x_1x_n^{n-2} \end{vmatrix}$$

Developing this determinant through the first column, we get

$$V = \begin{vmatrix} x_2 - x_1 & x_3 - x_1 & \cdots & x_n - x_1 \\ x_2^2 - x_1x_2 & x_3^2 - x_1x_3 & \cdots & x_n^2 - x_1x_n \\ x_2^3 - x_1x_2^2 & x_3^3 - x_1x_3^2 & \cdots & x_n^3 - x_1x_n^2 \\ \vdots & \vdots & & \vdots \\ x_2^{n-2} - x_1x_2^{n-3} & x_3^{n-2} - x_1x_3^{n-3} & \cdots & x_n^{n-2} - x_1x_n^{n-3} \\ x_2^{n-1} - x_1x_2^{n-2} & x_3^{n-1} - x_1x_3^{n-2} & \cdots & x_n^{n-1} - x_1x_n^{n-2} \end{vmatrix}$$

For $j = 1, 2, \dots, n-1$, we factor out $x_{j+1} - x_j$ from the j -th column. So

$$V = (x_2 - x_1)(x_3 - x_1) \cdots (x_n - x_1) \begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_2 & x_3 & \cdots & x_n \\ x_2^2 & x_3^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_2^{n-2} & x_3^{n-2} & \cdots & x_n^{n-2} \end{vmatrix}$$

This new determinant is the determinant of the Vandermonde matrix determined by x_2, x_3, \dots, x_n . So by induction,

$$\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_2 & x_3 & \cdots & x_n \\ x_2^2 & x_3^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_2^{n-2} & x_3^{n-2} & \cdots & x_n^{n-2} \end{vmatrix} = \prod_{2 \leq i < j \leq n} (x_j - x_i)$$

Hence

$$V = (x_2 - x_1)(x_3 - x_1) \cdots (x_n - x_1) \left(\prod_{2 \leq i < j \leq n} (x_j - x_i) \right) = \prod_{1 \leq i < j \leq n} (x_j - x_i) \quad \square$$

Corollary 7.5 Let x_1, x_2, \dots, x_n be n numbers. Then

$$\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix} \neq 0 \Leftrightarrow x_1, x_2, \dots, x_n \text{ are } n \text{ different numbers.}$$

Proof : By Proposition 7.4, we have that

$$\begin{vmatrix} 1 & 1 & \cdots & 1 \\ x_1 & x_2 & \cdots & x_n \\ x_1^2 & x_2^2 & \cdots & x_n^2 \\ \vdots & \vdots & & \vdots \\ x_1^{n-1} & x_2^{n-1} & \cdots & x_n^{n-1} \end{vmatrix} = \prod_{1 \leq i < j \leq n} (x_j - x_i)$$

Clearly

$$\prod_{1 \leq i < j \leq n} (x_j - x_i) = 0 \Leftrightarrow (x_j - x_i) = 0 \text{ for some } 1 \leq i < j \leq n \Leftrightarrow x_i = x_j \text{ for some } 1 \leq i < j \leq n$$

Hence $\prod_{1 \leq i < j \leq n} (x_j - x_i) \neq 0$ if and only if x_1, x_2, \dots, x_n are n different numbers. \square

7.3 Minimum Distance of BCH-Codes

In this section, we will show that the designed distance δ is a lower bound for the minimum distance of a BCH-code but is the actual minimum distance of a Reed-Solomon code.

Theorem 7.6 Let C be a BCH-code of length n over $GF(q)$ of designed distance δ . Then the minimum distance of C is at least δ .

Proof : We use the notations from Definition 7.1. Consider the matrix

$$H = \begin{bmatrix} 1 & \beta^b & \beta^{2b} & \cdots & \beta^{(n-1)b} \\ 1 & \beta^{b+1} & \beta^{2(b+1)} & \cdots & \beta^{(n-1)(b+1)} \\ 1 & \beta^{b+2} & \beta^{2(b+2)} & \cdots & \beta^{(n-1)(b+2)} \\ \vdots & \vdots & \vdots & & \vdots \\ 1 & \beta^{b+\delta-2} & \beta^{2(b+\delta-2)} & \cdots & \beta^{(n-1)(b+\delta-2)} \end{bmatrix}$$

First, we prove that $H\mathbf{c}^T = \mathbf{0}$ for all $\mathbf{c} \in C$. Let $\mathbf{c} = (a_0, a_1, \dots, a_{n-1}) \in C$. By Proposition 6.5(b), $\mathbf{c}(x) = g(x)f(x)$ for some $f(x) \in GF(q)[x]$. Fix $b \leq i \leq b + \delta - 2$. By definition of $g(x)$, we have that $m_{\beta^i}(x)$ divides $g(x)$. So $g(\beta^i) = 0$ since $m_{\beta^i}(\beta^i) = 0$. Hence $\mathbf{c}(\beta^i) = 0$. Thus

$$a_0 + a_1(\beta^i) + a_2(\beta^i)^2 + \cdots + a_{n-2}(\beta^i)^{n-2} + a_{n-1}(\beta^i)^{n-1} = 0$$

or

$$a_0 + a_1\beta^i + a_2\beta^{2i} + \cdots + a_{n-2}\beta^{(n-2)i} + a_{n-1}\beta^{(n-1)i} = 0$$

This is the dot-product of \mathbf{c} and the i -th row of H . Since this is true for $i = b, b+1, \dots, b+\delta-2$, we get that $H\mathbf{c}^T = \mathbf{0}$.

Next, we show that any $\delta - 1$ columns of H are linearly independent over $GF(q)$. We label the columns of H by C_0, C_1, \dots, C_{n-1} . Pick $\delta - 1$ columns of H , say columns $C_{j_1}, C_{j_2}, \dots, C_{j_{\delta-1}}$ where $0 \leq j_1 < j_2 < \cdots < j_{\delta-1} \leq n - 1$. Let H' be the submatrix of H formed by these $\delta - 1$ columns. Then

$$\det(H') = \begin{vmatrix} \beta^{j_1 b} & \beta^{j_2 b} & \cdots & \beta^{j_{\delta-1} b} \\ \beta^{j_1(b+1)} & \beta^{j_2(b+1)} & \cdots & \beta^{j_{\delta-1}(b+1)} \\ \beta^{j_1(b+2)} & \beta^{j_2(b+2)} & \cdots & \beta^{j_{\delta-1}(b+2)} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{j_1(b+\delta-2)} & \beta^{j_2(b+\delta-2)} & \cdots & \beta^{j_{\delta-1}(b+\delta-2)} \end{vmatrix}$$

For $i = 1, 2, \dots, \delta - 1$, we can factor out $\beta^{j_i b}$ from the i -th column. So

$$\det(H') = \beta^{b(j_1+j_2+\cdots+j_{\delta-1})} \begin{vmatrix} 1 & 1 & \cdots & 1 \\ \beta^{j_1} & \beta^{j_2} & \cdots & \beta^{j_{\delta-1}} \\ \beta^{2j_1} & \beta^{2j_2} & \cdots & \beta^{2j_{\delta-1}} \\ \vdots & \vdots & \ddots & \vdots \\ \beta^{(\delta-2)j_1} & \beta^{(\delta-2)j_2} & \cdots & \beta^{(\delta-2)j_{\delta-1}} \end{vmatrix}$$

Note that this involves the determinant of a Vandermonde matrix. Since β is an element of order n and $0 \leq j_1 < j_2 < \cdots < j_{\delta-1} \leq n - 1$, we have that $\beta^{j_1}, \beta^{j_2}, \dots, \beta^{j_{\delta-1}}$ are $\delta - 1$ different numbers. Hence $\det(H') \neq 0$ by Corollary 7.5. So the columns $C_{j_1}, C_{j_2}, \dots, C_{j_{\delta-1}}$ of H are linearly independent over $GF(q)$. Thus any $\delta - 1$ columns of H are linearly independent over $GF(q)$.

Finally, we show that $d(C) \geq \delta$. Suppose that $d(C) \leq \delta - 1$. Pick $\mathbf{0} \neq \mathbf{c} \in C$ with $w(\mathbf{c}) \leq \delta - 1$. Put $\mathbf{c} = (a_0, a_1, \dots, a_{n-1})$. Then $H\mathbf{c}^T = \mathbf{0}$. So

$$a_0 C_0 + a_1 C_1 + \cdots + a_{n-1} C_{n-1} = \mathbf{0}$$

Since $1 \leq w(\mathbf{c}) \leq \delta - 1$, this implies that some $\delta - 1$ columns of H are linearly dependent over $GF(q)$, a contradiction.

Hence $d(C) \geq \delta$. □

Corollary 7.7 *Let C be a Reed-Solomon code over $GF(q)$ of designed distance δ . Then the minimum distance of C is actually δ .*

Proof : Since a Reed-Solomon code is a special BCH-code, it follows from Theorem 7.6 that $\overline{d(C)} \geq \delta$. The Singleton Bound (Corollary 3.9) gives us that $n - k \geq d(C) - 1$. But $n = q - 1$ and $k = q - \delta$. So $\delta - 1 = (q - 1) - (q - \delta) \geq d(C) - 1$. Thus $\delta \geq d(C)$.

Hence $d(C) = \delta$. □

The next example illustrates that it is possible that the minimum distance of a BCH-code is bigger than the designed distance.

Example : We will construct a binary BCH-code of length 21. The cyclotomic cosets depending on $n = 21$ and $q = 2$ are

$$\{0\} , \{1, 2, 4, 8, 16, 11\} , \{3, 6, 12\} , \{5, 10, 20, 19, 17, 13\} , \{7, 14\} \text{ and } \{9, 18, 15\}$$

This corresponds to the following irreducible factors over $GF(2)$ of $x^{21} - 1$:

$$\begin{aligned} g_1(x) &= x - 1 \\ g_2(x) &= (x - \beta)(x - \beta^2)(x - \beta^4)(x - \beta^8)(x - \beta^{16})(x - \beta^{11}) \\ g_3(x) &= (x - \beta^3)(x - \beta^6)(x - \beta^{12}) \\ g_4(x) &= (x - \beta^5)(x - \beta^{10})(x - \beta^{20})(x - \beta^{19})(x - \beta^{17})(x - \beta^{13}) \\ g_5(x) &= (x - \beta^7)(x - \beta^{14}) \\ g_6(x) &= (x - \beta^9)(x - \beta^{18})(x - \beta^{15}) \end{aligned}$$

Let C be the binary BCH-code of length 21 and designed distance 4 with $b = 3$. So $d(C) \geq 4$ and the generator polynomial $g(x)$ of C is

$$g(x) = \text{lcm}(m_{\beta^3}(x), m_{\beta^4}(x), m_{\beta^5}(x)) = \text{lcm}(g_3(x), g_2(x), g_4(x)) = g_2(x)g_3(x)g_4(x)$$

Let C^* be the binary BCH-code of length 21 and designed distance 7 with $b = 1$. So $d(C^*) \geq 7$ and the generator polynomial $g^*(x)$ of C^* is

$$\begin{aligned} g(x) &= \text{lcm}(m_{\beta}(x), m_{\beta^2}(x), m_{\beta^3}(x)m_{\beta^4}(x), m_{\beta^5}(x), m_{\beta^6}(x)) \\ &= \text{lcm}(g_2(x), g_2(x), g_3(x), g_2(x), g_4(x), g_3(x)) \\ &= g_2(x)g_3(x)g_4(x) \end{aligned}$$

Since $g^*(x) = g(x)$, we have that $C = C^*$. Hence $d(C) = d(C^*) \geq 7$. ▷

7.4 Decoding BCH codes Using The PGZ-Algorithm

In this section, we describe a decoding algorithm by Peterson-Gorenstein-Zierler for the BCH-codes.

We start with a BCH-code. So let $GF(q)$ be a finite field, $n \in \mathbb{N}$ with $\text{gcd}(n, q) = 1$, $2 \leq \delta \leq n$, $b \geq 1$ and β an element of order n in some extension field $GF(q)$, and C the BCH-code of length n and designed distance δ with generator polynomial

$$g(x) = \text{lcm}(m_{\beta^b}(x), m_{\beta^{b+1}}(x), m_{\beta^{b+2}}(x), \dots, m_{\beta^{b+\delta-2}}(x))$$

Let t be maximal with $2t + 1 \leq \delta$. By Theorem 7.6, the minimum distance of C is at least δ . Hence C can correct at least t errors.

The algorithm we present will decode correctly if at most t errors occur during transmission. So suppose we transmit the codeword $\mathbf{c} \in C$ but we receive the word $\mathbf{y} \in GF(q)^n$. We put $\mathbf{y} = \mathbf{c} + \mathbf{e}$ where \mathbf{e} is the error vector. Assume that k errors occurred where $1 \leq k \leq t$ (so at

this point we assume that at least one error occurred). We will number the positions/entries of an element of $GF(q)^n$ from 0 to $n - 1$. Let $0 \leq i_1 < i_2 < \dots < i_k \leq n - 1$ be the positions of the errors and Y_1, Y_2, \dots, Y_k the size of the errors. Hence

$$\mathbf{e} = (0, \dots, 0, \underbrace{Y_1}_{\text{position } i_1}, 0, \dots, 0, \underbrace{Y_2}_{\text{position } i_2}, 0, \dots, 0, \underbrace{Y_k}_{\text{position } i_k}, 0, \dots, 0)$$

We now introduce the very important concept of syndromes. For $1 \leq j \leq \delta - 1$, we define the j -th syndrome of \mathbf{y} (notation : S_j) as

$$S_j = \mathbf{y}(\beta^{b+j-1}) \quad \text{for } 1 \leq j \leq \delta - 1$$

Fix $1 \leq j \leq \delta - 1$. Then $m_{\beta^{b+j}}(x)$ divides $g(x)$. But $g(x)$ divides $\mathbf{c}(x)$ since $\mathbf{c} \in C$. Hence $m_{\beta^{b+j}}(x)$ divides $\mathbf{c}(x)$. Since $m_{\beta^{b+j}}(\beta^{b+j}) = 0$, we have that $\mathbf{c}(\beta^{b+j}) = 0$. So

$$\begin{aligned} S_j &= \mathbf{y}(\beta^{b+j}) \\ &= (\mathbf{c} + \mathbf{e})(\beta^{b+j}) \\ &= \mathbf{c}(\beta^{b+j}) + \mathbf{e}(\beta^{b+j}) \\ &= \mathbf{e}(\beta^{b+j}) \\ &= Y_1(\beta^{b+j-1})^{i_1} + Y_2(\beta^{b+j-1})^{i_2} + \dots + Y_k(\beta^{b+j-1})^{i_k} \\ &= Y_1(\beta^{i_1})^{b+j-1} + Y_2(\beta^{i_2})^{b+j-1} + \dots + Y_k(\beta^{i_k})^{b+j-1} \quad \text{for } 1 \leq j \leq \delta - 1 \end{aligned}$$

We put

$$\beta^{i_l} = X_l \quad \text{for } l = 1, 2, \dots, k$$

Then

$$S_j = Y_1 X_1^{b+j-1} + Y_2 X_2^{b+j-1} + \dots + Y_k X_k^{b+j-1} = \sum_{i=1}^k Y_i X_i^{b+j-1} \quad \text{for } 1 \leq j \leq \delta - 1 \quad (*)$$

Notice that this is a system of at least $2t$ equations. The only known quantities are the syndromes $S_1, S_2, \dots, S_{\delta-1}$. At this point, $k, X_1, X_2, \dots, X_k, Y_1, Y_2, \dots, Y_k$ are unknown.

Next, we introduce another important concept. We define the *error-locating polynomial* (notation $s(x)$) as

$$s(x) = (1 - X_1 x)(1 - X_2 x) \cdots (1 - X_k x)$$

Since $s(x)$ is a polynomial of degree k , we can write

$$s(x) = 1 + s_1 x + s_2 x^2 + \dots + s_k x^k$$

We put $s_0 = 1$. We will deduce a system of linear equations in s_1, s_2, \dots, s_k . Once k and s_1, s_2, \dots, s_k are known, we will be able to decode correctly: since we know $s(x)$, we can find its roots (so we know X_1, X_2, \dots, X_k) and hence the positions i_1, i_2, \dots, i_k in which the errors occur; finally we solve (*) to find the size of the errors Y_1, Y_2, \dots, Y_k .

By definition of the error-locating polynomial, we have that $s(X_i^{-1}) = 0$ for $1 \leq i \leq k$. Hence for $1 \leq i, j \leq k$, we have that

$$\begin{aligned}
0 &= Y_i X_i^{b+j-1+k} s(X_i^{-1}) \\
&= Y_i X_i^{b+j-1+k} (s_0 + s_1 X_i^{-1} + s_2 X_i^{-2} + \cdots + s_k X_i^{-k}) \\
&= Y_i (s_0 X_i^{b+j-1+k} + s_1 X_i^{b+j-2+k} + s_2 X_i^{b+j-3+k} + \cdots + s_k X_i^{b+j-1}) \\
&= Y_i \left(\sum_{l=0}^k s_l X_i^{b+j-1+k-l} \right)
\end{aligned}$$

Summing over i and switching the order of summation, we find that

$$0 = \sum_{i=1}^k Y_i \left(\sum_{l=0}^k s_l X_i^{b+j-1+k-l} \right) = \sum_{l=0}^k s_l \left(\sum_{i=1}^k Y_i X_i^{b+j-1+k-l} \right) = \sum_{l=0}^k s_l S_{j+k-l}$$

Since $s_0 = 1$, we can rewrite this as

$$\sum_{l=1}^k s_l S_{j+k-l} = -S_{j+k} \quad \text{for } j = 1, 2, \dots, k$$

This is a system of k linear equations in the k unknowns s_1, s_2, \dots, s_k . Rewriting this in matrix form we get

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_k \\ S_2 & S_3 & \cdots & S_{k+1} \\ \vdots & \vdots & & \vdots \\ S_k & S_{k+1} & \cdots & S_{2k-1} \end{bmatrix} \begin{bmatrix} s_k \\ s_{k-1} \\ \vdots \\ s_1 \end{bmatrix} = \begin{bmatrix} -S_{k+1} \\ -S_{k+2} \\ \vdots \\ -S_{2k} \end{bmatrix}$$

The only problem left at this point is to figure out the value of k . The following property solves this problem.

Proposition 7.8 *With the above notations, put*

$$P_d = \begin{bmatrix} S_1 & S_2 & \cdots & S_d \\ S_2 & S_3 & \cdots & S_{d+1} \\ \vdots & \vdots & & \vdots \\ S_d & S_{d+1} & \cdots & S_{2d-1} \end{bmatrix}$$

for $d = 1, \dots, t$. If $d > k$ then $\det(P_d) = 0$. If $d = k$, then $\det(P_d) \neq 0$.

Proof : Let $k \leq d \leq t$. Put $X_j = Y_j = 0$ for $k+1 \leq j \leq t$. Put

$$Q = \begin{bmatrix} 1 & 1 & \cdots & 1 \\ X_1 & X_2 & \cdots & X_d \\ X_1^2 & X_2^2 & \cdots & X_d^2 \\ \vdots & \vdots & & \vdots \\ X_1^{d-1} & X_2^{d-1} & \cdots & X_d^{d-1} \end{bmatrix}$$

and

$$D = \begin{bmatrix} X_1^b Y_1 & 0 & \cdots & 0 \\ 0 & X_2^b Y_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & X_d^b Y_d \end{bmatrix}$$

We number the rows and columns of P_d , Q and D from 1 to d . Let $1 \leq i, j \leq d$. We get that

$$\begin{aligned} (QDQ^T)_{ij} &= \sum_{l,s=1}^d Q_{il} D_{ls} (Q^T)_{sj} \\ &= \sum_{l=1}^d Q_{il} D_{ll} (Q^T)_{lj} && \text{since } D_{ls} = 0 \text{ if } s \neq l \\ &= \sum_{l=1}^d Q_{il} X_l^b Y_l Q_{jl} \\ &= \sum_{l=1}^d X_l^{i-1} X_l^b Y_l X_l^{j-1} \\ &= \sum_{l=1}^k Y_l X_l^{b+i+j-2} \\ &= \sum_{l=1}^k Y_l X_l^{b+i+j-2} && \text{since } X_l = Y_l = 0 \text{ if } l > k \\ &= S_{i+j-1} \\ &= (P_d)_{ij}. \end{aligned}$$

So $P_d = QDQ^T$. Hence

$$\det(P) = \det(QDQ^T) = \det(Q) \det(D) \det(Q^T) = \det(Q) \det(D) \det(Q) = \det(Q)^2 \det(D)$$

Since D is a diagonal matrix, we get that

$$\det(D) = (X_1^b Y_1)(X_2^b Y_2) \cdots (X_d^b Y_d)$$

Suppose first that $d > k$. Then $X_d = Y_d = 0$ and so $\det(D) = 0$. Hence $\det(P_d) = 0$.

Suppose next that $d = k$. Since $X_j = \beta^{ij}$, we get that $X_j \neq 0$ for $j = 1, 2, \dots, k$. Moreover, since Y_1, Y_2, \dots, Y_k are the sizes of the errors, we have that $Y_j \neq 0$ for $j = 1, 2, \dots, k$. So $\det(D) \neq 0$. Note that X_1, X_2, \dots, X_k are k different numbers since we assumed that exactly k errors occurred. Since Q is a Vandermonde matrix defined by X_1, X_2, \dots, X_k , it follows from Corollary 7.5 that $\det(Q) \neq 0$. Hence $\det(P_d) \neq 0$. \square

This property allows us to find the value of k assuming that at least one and at most t errors occurred : k is the largest value of d for which $\det(P_d) \neq 0$. We can find k by evaluating $\det(P_d)$ for $d = t, t-1, t-2, \dots$ until we find a non-zero value.

The decoding algorithm we described here assumed that at least one error and at most t errors occurred during transmission. What if no errors occurred? Again, the syndromes help us out as the following lemma shows.

Lemma 7.9 *Assume that (with the above notations), $S_j = 0$ for $j = 1, 2, \dots, \delta - 1$. Then $\mathbf{y} \in C$.*

Proof : For $1 \leq j \leq \delta - 1$, we have that $S_j = \mathbf{y}(\beta^{b+j-1}) = 0$ and so $m_{\beta^{b+j-1}}(x)$ divides $\mathbf{y}(x)$. Hence $g(x) = \text{lcm}(m_{\beta^b}(x), m_{\beta^{b+1}}(x), \dots, m_{\beta^{b+\delta-2}}(x))$ divides $\mathbf{y}(x)$. So $\mathbf{y} \in C$. \square

Now we can present the complete Peterson-Gorenstein-Zierler decoding algorithm. Make sure to understand that we will decode correctly if at most t errors occur during transmission.

1. Upon receiving the word \mathbf{y} , evaluate the syndromes $S_1, S_2, \dots, S_{\delta-1}$ (recall that $S_j = \mathbf{y}(\beta^{b+j-1})$ for all j). If $S_1 = S_2 = \dots = S_{\delta-1} = 0$ then \mathbf{y} is a codeword and we decode as \mathbf{y} ; otherwise proceed to step 2.
2. Find the largest integer k such that $1 \leq k \leq t$ and

$$\begin{vmatrix} S_1 & S_2 & \cdots & S_k \\ S_2 & S_3 & \cdots & S_{k+1} \\ \vdots & \vdots & & \vdots \\ S_k & S_{k+1} & \cdots & S_{2k-1} \end{vmatrix} \neq 0$$

3. Solve

$$\begin{bmatrix} S_1 & S_2 & \cdots & S_k \\ S_2 & S_3 & \cdots & S_{k+1} \\ \vdots & \vdots & & \vdots \\ S_k & S_{k+1} & \cdots & S_{2k-1} \end{bmatrix} \begin{bmatrix} s_k \\ s_{k-1} \\ \vdots \\ s_1 \end{bmatrix} = \begin{bmatrix} -S_{k+1} \\ -S_{k+2} \\ \vdots \\ -S_{2k} \end{bmatrix}$$

for s_1, s_2, \dots, s_k .

4. Put $s(x) = 1 + s_1x + s_2x^2 + \dots + s_kx^k$. Find $0 \leq i_1 < i_2 < \dots < i_k \leq n - 1$ such that $X_1 = \beta^{i_1}, X_2 = \beta^{i_2}, \dots, X_k = \beta^{i_k}$ are the reciprocals of the zeros of $s(x)$.

5. Solve

$$\begin{bmatrix} X_1^b & X_2^b & \cdots & X_k^b \\ X_1^{b+1} & X_2^{b+1} & \cdots & X_k^{b+1} \\ \vdots & \vdots & & \vdots \\ X_1^{b+\delta-2} & X_2^{b+\delta-2} & \cdots & X_k^{b+\delta-2} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_k \end{bmatrix} = \begin{bmatrix} S_1 \\ S_2 \\ \vdots \\ S_{\delta-1} \end{bmatrix}$$

for Y_1, \dots, Y_k .

6. Put

$$\mathbf{e} = (0, \dots, 0, \underbrace{Y_1}_{\text{position } i_1}, 0, \dots, 0, \underbrace{Y_2}_{\text{position } i_2}, 0, \dots, 0, \underbrace{Y_k}_{\text{position } i_k}, 0, \dots, 0)$$

and decode as $\mathbf{c} := \mathbf{y} - \mathbf{e}$.

If at any point we can not complete a step, we declare a decoding error. Note that a decoding error implies that more than t errors occurred.

We might not be able to complete a step because

2. $\det(P_d) = 0$ for $d = t, t - 1, \dots, 2, 1$.
4. $s(x)$ does not have k different roots or the roots are not powers of β .
5. The system of equations (*) does not have a unique solution or has a unique solution but $Y_i = 0$ for some $1 \leq i \leq k$.

We illustrate this decoding algorithm with two examples.

Examples

1. A binary BCH-code of length $n = 21$ and designed distance $\delta = 7$.

Let α be a primitive element of $GF(64)$ as given in our table. Put $\beta = \alpha^3$. Then β is an element of order 21. Finally, put $\delta = 7$ and $b = 1$. Then our code C has generator polynomial

$$g(x) = \text{lcm}(m_\beta(x), m_{\beta^2}(x), m_{\beta^3}(x), m_{\beta^4}(x), m_{\beta^5}(x), m_{\beta^6}(x))$$

where $m_{\beta^i}(x)$ is the minimal polynomial of β^i over $GF(2)$.

Suppose we receive the word

$$\mathbf{y} = 001001000000000000000000$$

So $\mathbf{y}(x) = x^2 + x^5$.

First, we calculate $\delta - 1 = 6$ syndromes. We get

$$\begin{aligned} S_1 &= \mathbf{y}(\beta) = \mathbf{y}(\alpha^3) = \alpha^6 + \alpha^{15} = 000011 + 101000 = 101011 = \alpha^{51} \\ S_2 &= \mathbf{y}(\beta^2) = \mathbf{y}(\alpha^6) = \alpha^{12} + \alpha^{30} = 000101 + 110011 = 110110 = \alpha^{39} \\ S_3 &= \mathbf{y}(\beta^3) = \mathbf{y}(\alpha^9) = \alpha^{18} + \alpha^{45} = 001111 + 011001 = 010110 = \alpha^{36} \\ S_4 &= \mathbf{y}(\beta^4) = \mathbf{y}(\alpha^{12}) = \alpha^{24} + \alpha^{60} = 010001 + 111001 = 101000 = \alpha^{15} \\ S_5 &= \mathbf{y}(\beta^5) = \mathbf{y}(\alpha^{15}) = \alpha^{30} + \alpha^{75} = \alpha^{30} + \alpha^{12} = 110011 + 000101 = 110110 = \alpha^{39} \\ S_6 &= \mathbf{y}(\beta^6) = \mathbf{y}(\alpha^{18}) = \alpha^{36} + \alpha^{90} = \alpha^{36} + \alpha^{27} = 010110 + 001110 = 011000 = \alpha^9 \end{aligned}$$

Next, we calculate k , the number of errors that occurred. Since $\delta = 7$, we have that $t = 3$.

So we calculate (using Sarrus' rule)

$$\begin{aligned} \begin{vmatrix} \alpha^{51} & \alpha^{39} & \alpha^{36} \\ \alpha^{39} & \alpha^{36} & \alpha^{15} \\ \alpha^{36} & \alpha^{15} & \alpha^{39} \end{vmatrix} &= \alpha^{51}\alpha^{36}\alpha^{39} + \alpha^{39}\alpha^{15}\alpha^{36} + \alpha^{39}\alpha^{15}\alpha^{36} + \alpha^{36}\alpha^{36}\alpha^{36} + \alpha^{39}\alpha^{39}\alpha^{39} + \alpha^{15}\alpha^{15}\alpha^{51} \\ &= \alpha^{126} + \alpha^{108} + \alpha^{117} + \alpha^{81} \\ &= 1 + \alpha^{45} + \alpha^{54} + \alpha^{18} \\ &= 000001 + 011001 + 010111 + 001111 \\ &= 000000 \\ &= 0 \end{aligned}$$

So $k \neq 3$. We continue with

$$\begin{vmatrix} \alpha^{51} & \alpha^{39} \\ \alpha^{39} & \alpha^{36} \end{vmatrix} = \alpha^{51}\alpha^{36} + \alpha^{39}\alpha^{39} = \alpha^{87} + \alpha^{78} = \alpha^{24} + \alpha^{15} = 010001 + 101000 = 111001 = \alpha^{60} \neq 0$$

Hence we assume that $k = 2$: two errors occurred.

Next, we find the error-locating polynomial. So we have to solve

$$\begin{bmatrix} \alpha^{51} & \alpha^{39} \\ \alpha^{39} & \alpha^{36} \end{bmatrix} \begin{bmatrix} s_2 \\ s_1 \end{bmatrix} = \begin{bmatrix} \alpha^{36} \\ \alpha^{15} \end{bmatrix}$$

We use Cramer's Rule.

Since

$$\begin{vmatrix} \alpha^{36} & \alpha^{39} \\ \alpha^{15} & \alpha^{36} \end{vmatrix} = \alpha^{36}\alpha^{36} + \alpha^{15}\alpha^{39} = \alpha^{72} + \alpha^{54} = \alpha^9 + \alpha^{54} = 011000 + 010111 = 001111 = \alpha^{18}$$

we get that

$$s_2 = \frac{\alpha^{18}}{\alpha^{60}} = \alpha^{-42} = \alpha^{21}$$

Since

$$\begin{vmatrix} \alpha^{51} & \alpha^{36} \\ \alpha^{39} & \alpha^{15} \end{vmatrix} = \alpha^{51}\alpha^{15} + \alpha^{36}\alpha^{39} = \alpha^{66} + \alpha^{75} = \alpha^3 + \alpha^{12} = 001000 + 000101 = 001101 = \alpha^{48}$$

we get that

$$s_1 = \frac{\alpha^{48}}{\alpha^{60}} = \alpha^{-12} = \alpha^{51}$$

So the error-locating polynomial is

$$s(x) = 1 + s_1x + s_2x = 1 + \alpha^{51}x + \alpha^{21}x^2$$

Next, we need to find the roots of $s(x)$. We should find two roots and they should be powers of β . Since we really want the reciprocals of the roots, we start

$$\begin{aligned} s(\beta^0) &= s(1) = 1 + \alpha^{51} + \alpha^{21} = 000001 + 101011 + 111011 = 010001 \neq 0 \\ s(\beta^{-1}) &= s(\alpha^{-3}) = 1 + \alpha^{51}\alpha^{-3} + \alpha^{21}\alpha^{-6} = 1 + \alpha^{48} + \alpha^{15} = 000001 + 001101 + 101000 = 100100 \neq 0 \\ s(\beta^{-2}) &= s(\alpha^{-6}) = 1 + \alpha^{51}\alpha^{-6} + \alpha^{21}\alpha^{-12} = 1 + \alpha^{45} + \alpha^9 = 000001 + 011001 + 011000 = 000000 = 0 \end{aligned}$$

So $\beta^{-2} = \alpha^{-6}$ is a root of $s(x)$. Since the product of the roots of $s(x)$ is α^{-21} , we get that the second root of $s(x)$ is $\alpha^{-15} = \beta^{-5}$.

Hence

$$X_1 = \beta^2 = \alpha^6 \quad \text{and} \quad X_2 = \beta^5 = \alpha^{15}$$

That means that the first error occurred in the second position while the second error occurred in the fifth position (recall that we start numbering the position from zero).

Finally, we find the error-sizes Y_1 and Y_2 . Since we are working binary, we should find $Y_1 = Y_2 = 1$. We have to solve

$$\begin{bmatrix} \alpha^6 & \alpha^{15} \\ \alpha^{12} & \alpha^{30} \\ \alpha^{18} & \alpha^{45} \\ \alpha^{24} & \alpha^{60} \\ \alpha^{30} & \alpha^{75} \\ \alpha^{36} & \alpha^{90} \end{bmatrix} \begin{bmatrix} Y_1 \\ Y_2 \end{bmatrix} = \begin{bmatrix} \alpha^{51} \\ \alpha^{39} \\ \alpha^{36} \\ \alpha^{15} \\ \alpha^{39} \\ \alpha^9 \end{bmatrix}$$

A quick check does indeed show that $Y_1 = Y_2 = 1$. For example:

$$\begin{aligned} \alpha^6 + \alpha^{15} &= 000011 + 101000 = 101011 = \alpha^{51} \\ \alpha^{18} + \alpha^{45} &= 001111 + 011001 = 010110 = \alpha^{36} \end{aligned}$$

So the error vector is

$$\mathbf{e} = 001001000000000000000000$$

Hence we decode \mathbf{y} as

$$\mathbf{y} - \mathbf{e} = \mathbf{0} = 000000000000000000000000$$

Note that we could have seen this of course in the beginning: since $d(\mathbf{y}, \mathbf{0}) = 2 \leq 3$ and the code is at least three-error correcting, we will have to decode \mathbf{y} as $\mathbf{0}$. We went through the algorithm to illustrate the decoding process.

2. A BCH-code over $GF(4)$ of length $n = 5$ and designed distance $\delta = 3$.

Let α be a primitive element of $GF(16)$ as given in our table. Then $GF(4) = \{0, 1, \alpha^5, \alpha^{10}\}$ is a subfield of $GF(16)$. Put $\beta = \alpha^3$. Then β is an element of order 5. Finally, put $\delta = 3$ and $b = 2$. Then our code C has generator polynomial

$$g(x) = \text{lcm}(m_{\beta^2}(x), m_{\beta^3}(x))$$

where $m_{\beta^i}(x)$ is the minimal polynomial of β^i over $GF(4)$.

Suppose first we receive

$$\mathbf{y} = 110\alpha^5 0$$

So $\mathbf{y}(x) = 1 + x + \alpha^5 x^3$.

First, we calculate $\delta - 1 = 2$ syndromes. We get

$$\begin{aligned} S_1 = \mathbf{y}(\beta^2) = \mathbf{y}(\alpha^6) &= 1 + \alpha^6 + \alpha^{23} = 1 + \alpha^6 + \alpha^8 = 0001 + 1111 + 1110 = 0000 = 0 \\ S_2 = \mathbf{y}(\beta^3) = \mathbf{y}(\alpha^9) &= 1 + \alpha^9 + \alpha^{32} = 1 + \alpha^9 + \alpha^2 = 0001 + 0101 + 0100 = 0000 = 0 \end{aligned}$$

Hence \mathbf{y} is a codeword and we decode as \mathbf{y} .

Suppose next we receive

$$\mathbf{y} = 110\alpha^{10}1$$

So $\mathbf{y}(x) = 1 + x + \alpha^{10}x^3 + x^4$.

First we calculate the two syndromes. We get

$$\begin{aligned} S_1 = \mathbf{y}(\beta^2) = \mathbf{y}(\alpha^6) &= 1 + \alpha^6 + \alpha^{28} + \alpha^{24} = 1 + \alpha^6 + \alpha^{13} + \alpha^9 = 0001 + 1111 + 0110 + 0101 = 1101 = \alpha^{11} \\ S_2 = \mathbf{y}(\beta^3) = \mathbf{y}(\alpha^9) &= 1 + \alpha^9 + \alpha^{37} + \alpha^{36} = 1 + \alpha^9 + \alpha^7 + \alpha^6 = 0001 + 0101 + 0111 + 1111 = 1100 = \alpha^{14} \end{aligned}$$

Next, we calculate k , the number of errors that occurred. Since $\delta = 3$, we have that $t = 1$.

So we ‘calculate’

$$|\alpha^{11}| = \alpha^{11} \neq 0$$

Hence we assume that $k = 1$: one errors occurred.

Next, we find the error-locating polynomial. So we have to solve

$$\alpha^{11}s_1 = \alpha^{14}$$

Hence

$$s_1 = \frac{\alpha^{14}}{\alpha^{11}} = \alpha^3$$

So the error-locating polynomial is

$$s(x) = 1 + s_1x = 1 + \alpha^3x$$

Next, we need to find the roots of $s(x)$. We should find one roots and it should be powers of β . Since $\deg(s(x)) = 1$, we easily get that

$$x = \frac{1}{\alpha^3} = \alpha^{-3}$$

So

$$X_1 = \alpha^3 = \beta^1$$

That means that the error occurred in the first position (recall that we start numbering the position from zero).

Finally, we find the error-size Y_1 . We have to solve

$$\begin{bmatrix} \alpha^6 \\ \alpha^9 \end{bmatrix} [Y_1] = \begin{bmatrix} \alpha^{11} \\ \alpha^{14} \end{bmatrix}$$

We easily get that

$$Y_1 = \frac{\alpha^{11}}{\alpha^6} = \frac{\alpha^{14}}{\alpha^9} = \alpha^5$$

So the error vector is

$$\mathbf{e} = 0\alpha^5000$$

Hence we decode \mathbf{y} as

$$\mathbf{y} - \mathbf{e} = 110\alpha^{10}1 - 0\alpha^5000 = 1\alpha^{10}0\alpha^{10}1$$

since $1 + \alpha^5 = 0001 + 1011 = 1010 = \alpha^{10}$.

▷

7.5 Euclidean Algorithm

In section, we illustrate an implementation of the Euclidean Algorithm. This will be used in the next section in a different decoding algorithm for the BCH-codes.

The following property is the backbone of the Euclidean Algorithm for polynomials over fields:

$$\text{If } a(x) = b(x)q(x) + r(x) \text{ then } \gcd(a(x), b(x)) = \gcd(b(x), r(x)).$$

So let $a(x), b(x)$ be polynomials over a field with $\deg(a(x)) \geq \deg(b(x))$ and $b(x) \neq 0$. We define the degree of the zero polynomial to be $-\infty$. Then the following algorithm will result in $\gcd(a(x), b(x))$ and write it as a linear combination of $a(x)$ and $b(x)$.

The algorithm will produce an augmented matrix of the form

$$\left[\begin{array}{cc|c} 1 & 0 & a(x) \\ 0 & 1 & b(x) \\ u_1(x) & v_1(x) & r_1(x) \\ u_2(x) & v_2(x) & r_2(x) \\ \vdots & \vdots & \vdots \end{array} \right]$$

We put

$$\begin{aligned} r_{-1}(x) &= a(x) & r_0(x) &= b(x) \\ u_{-1}(x) &= 1 & u_0(x) &= 0 \\ v_{-1}(x) &= 0 & v_0(x) &= 1 \end{aligned}$$

At the i -th step (for $i \geq 1$), if $r_{i-1}(x) \neq 0$ we use the Division Algorithm to divide $r_{i-2}(x)$ by $r_{i-1}(x)$ and write

$$r_{i-2}(x) = r_{i-1}(x)q_i(x) + r_i(x) \quad \text{with } \deg(r_i(x)) < \deg(r_{i-1}(x))$$

We define

$$u_i(x) = u_{i-2}(x) - q_i(x)u_{i-1}(x) \quad \text{and} \quad v_i(x) = v_{i-2}(x) - q_i(x)v_{i-1}(x)$$

The algorithm ends when $r_n(x) \equiv 0$. Then we have that

$$\boxed{\gcd(a(x), b(x)) = r_{n-1}(x) = u_{n-1}(x)a(x) + v_{n-1}(x)b(x)}$$

We illustrate this algorithm with two examples.

Examples:

1. $a(x) = x^5 + 1$ and $b(x) = x^3 + 1$ over $GF(2)$. The matrix looks like

$$\begin{array}{cccc} 1 & 0 & x^5 + 1 & \\ 0 & 1 & x^3 + 1 & x^5 + 1 = (x^3 + 1)x^2 + (x^2 + 1) \quad R_3 = R_1 - x^2 R_2 \\ 1 & x^2 & x^2 + 1 & x^3 + 1 = (x^2 + 1)x + (x + 1) \quad R_4 = R_2 - x R_3 \\ x & x^3 + 1 & x + 1 & x^2 + 1 = (x + 1)(x + 1) \quad \text{so we stop} \end{array}$$

$$\boxed{\gcd(x^5 + 1, x^3 + 1) = x + 1 = x(x^5 + 1) + (x^3 + 1)(x^3 + 1)}$$

2. $a(x) = x^5$ and $b(x) = x^3 + x + \alpha^5$ over $GF(8)$. The matrix looks like

$$\begin{array}{ccc} 1 & 0 & x^5 \\ 0 & 1 & x^3 + x + \alpha^5 \quad (1) \\ 1 & x^2 + 1 & \alpha^5 x^2 + x + \alpha^5 \quad (2) \\ \alpha^2 x + \alpha^4 & \alpha^2 x^3 + \alpha^4 x^2 + \alpha^2 x + \alpha^5 & \alpha^4 x + \alpha^3 \quad (3) \\ \alpha^3 x^2 + \alpha^2 x + \alpha^4 & \alpha^3 x^4 + \alpha^2 x^3 + \alpha^6 x^2 + \alpha^4 x + \alpha^2 & 1 \quad (4) \end{array}$$

because

$$\begin{array}{ll} (1) & x^5 = (x^3 + x + \alpha^5)(x^2 + 1) + (\alpha^5 x^2 + x + \alpha^5) & \text{so } R_3 = R_1 - (x^2 + 1)R_2 \\ (2) & x^3 + x + \alpha^5 = (\alpha^5 x^2 + x + \alpha^5)(\alpha^2 x + \alpha^4) + (\alpha^4 x + \alpha^3) & \text{so } R_4 = R_2 - (\alpha^2 x + \alpha^4)R_3 \\ (3) & \alpha^5 x^2 + x + \alpha^5 = (\alpha^4 x + \alpha^3)(\alpha x + \alpha) + 1 & \text{so } R_5 = R_3 - (\alpha x + \alpha)R_4 \\ (4) & \alpha^4 x + \alpha^3 = 1 \cdot (\alpha^4 x + \alpha^3) + 0 & \text{so we stop} \end{array}$$

$$\boxed{\gcd(x^5, x^3 + x + \alpha^5) = 1 = (\alpha^3 x^2 + \alpha^2 x + \alpha^4)x^5 + (\alpha^3 x^4 + \alpha^2 x^3 + \alpha^6 x^2 + \alpha^4 x + \alpha^2)(x^3 + x + \alpha^5)}$$

We finish this section by proving some general properties about $u_i(x)$, $v_i(x)$ and $r_i(x)$ (see page 62 for the definition of these terms).

Proposition 7.10 *Let $a(x)$ and $b(x)$ be the polynomials used in the Euclidean Algorithm. Then the following holds:*

- (a) $u_i(x)a(x) + v_i(x)b(x) = r_i(x)$ for all $i \geq -1$
- (b) $u_{i-1}(x)v_i(x) - u_i(x)v_{i-1}(x) = (-1)^i$ for all $i \geq 0$
- (c) $\gcd(u_i(x), v_i(x)) = 1$ for all $i \geq 0$
- (d) $q_i(x) \neq 0$ for all $i \geq 1$.
- (e) $\deg(v_i(x)) = \deg(q_i(x)v_{i-1}(x))$ for all $i \geq 1$
- (f) $\deg(v_i(x)) \geq \deg(v_{i-1}(x))$ for all $i \geq 0$.
- (g) $\deg(v_i(x)) + \deg(r_{i-1}(x)) = \deg(a(x))$ for all $i \geq 0$

Proof : (a) We induct on i .
For $i = -1$, we have that

$$u_{-1}(x)a(x) + v_{-1}(x)b(x) = 1 \cdot a(x) + 0 \cdot b(x) = a(x) = r_{-1}(x)$$

For $i = 0$, we have that

$$u_0(x)a(x) + v_0(x)b(x) = 0 \cdot a(x) + 1 \cdot b(x) = b(x) = r_0(x)$$

Assume that $i > 0$. Then by induction, we get that

$$\begin{aligned}
u_i(x)a(x) + v_i(x)b(x) &= (u_{i-2}(x) - q_i(x)u_{i-1}(x))a(x) + (v_{i-2}(x) - q_i(x)v_{i-1}(x))b(x) \\
&= [u_{i-2}(x)a(x) + v_{i-2}(x)b(x)] - q_i(x)[u_{i-1}(x)a(x) + v_{i-1}(x)b(x)] \\
&= r_{i-2}(x) - q_i(x)r_{i-1}(x) \\
&= r_i(x)
\end{aligned}$$

(b) We induct on i .

For $i = 0$, we have that

$$u_{-1}(x)v_0(x) - u_0(x)v_{-1}(x) = 1 \cdot 1 - 0 \cdot 0 = 1 = 1^0$$

Assume that $i > 0$. Then by induction, we get that

$$\begin{aligned}
u_{i-1}(x)v_i(x) - u_i(x)v_{i-1}(x) &= u_{i-1}(x)(v_{i-2}(x) - q_i(x)v_{i-1}(x)) - (u_{i-2}(x) - q_i(x)u_{i-1}(x))v_{i-1}(x) \\
&= -(u_{i-2}(x)v_{i-1}(x) - u_{i-1}(x)v_{i-2}(x)) \\
&= -(-1)^{i-1} \\
&= (-1)^i
\end{aligned}$$

(c) Let $i \geq 0$. Then it follows from (b) that

$$u_{i-1}(x)v_i(x) + (-v_{i-1}(x))u_i(x) = (-1)^i$$

and so

$$[(-1)^i u_{i-1}(x)]v_i(x) + [(-1)^{i-1} v_{i-1}(x)]u_i(x) = 1$$

Thus 1 a linear combination of $u_i(x)$ and $v_i(x)$. Hence $\gcd(u_i(x), v_i(x)) = 1$.

(d) Pick $i \geq 1$. Since $r_{i-2}(x) = r_{i-1}(x)q_i(x) + r_i(x)$ and $\deg(r_i(x)) < \deg(r_{i-1}(x)) \leq \deg(r_{i-2}(x))$, we have that $q_i(x) \neq 0$.

(e) We induct on i .

For $i = 1$, we have that

$$\deg(v_1(x)) = \deg(v_{-1}(x) - q_1(x)v_0(x)) = \deg(0 - q_1(x)v_0(x)) = \deg(q_1(x)v_0(x))$$

Assume that $i \geq 2$. Recall that $r_{i-2}(x) = r_{i-1}(x)q_i(x) + r_i(x)$. Since

$$\deg(r_i(x)) < \deg(r_{i-1}(x)) < \deg(r_{i-2}(x))$$

we must have that $\deg(q_i(x)) \geq 1$. So by induction and (d), we have that

$$\deg(q_i(x)v_{i-1}(x)) > \deg(v_{i-1}(x)) = \deg(q_{i-1}(x)v_{i-2}(x)) \geq \deg(v_{i-2}(x))$$

Hence we get that

$$\deg(v_i(x)) = \deg(v_{i-2}(x) - q_i(x)v_{i-1}(x)) = \deg(q_i(x)v_{i-1}(x))$$

(f) If $i = 0$ then

$$\deg(v_0(x)) = \deg(1) = 0 \geq -\infty = \deg(0) = \deg(v_{-1}(x))$$

If $i > 0$, then it follows from (e) and (d) that

$$\deg(v_i(x)) = \deg(q_i(x)v_{i-1}(x)) = \deg(q_i(x)) + \deg(v_{i-1}(x)) \geq \deg(v_{i-1}(x))$$

(g) We induct on i .

For $i = 0$, we have that $v_0(x) = 1$ and $r_{-1}(x) = a(x)$. Hence

$$\deg(v_0(x)) + \deg(r_{-1}(x)) = 0 + \deg(a(x)) = \deg(a(x))$$

Assume that $i \geq 1$. Recall that

$$r_{i-2}(x) = r_{i-1}(x)q_i(x) + r_i(x)$$

Since $\deg(r_i(x)) < \deg(r_{i-1}(x))$ and $q_i(x) \not\equiv 0$, we have that

$$\deg(r_{i-2}(x)) = \deg(r_{i-1}(x)q_i(x)) = \deg(r_{i-1}(x)) + \deg(q_i(x))$$

and so

$$\deg(r_{i-1}(x)) = \deg(r_{i-2}(x)) - \deg(q_i(x))$$

By induction we have that

$$\deg(v_{i-1}(x)) + \deg(r_{i-2}(x)) = \deg(a(x))$$

and so

$$\deg(r_{i-2}(x)) = \deg(a(x)) - \deg(v_{i-1}(x))$$

Hence

$$\deg(r_{i-1}(x)) = \deg(r_{i-2}(x)) - \deg(q_i(x)) = \deg(a(x)) - \deg(v_{i-1}(x)) - \deg(q_i(x))$$

By (e), we have that

$$\deg(v_i(x)) = \deg(q_i(x)v_{i-1}(x)) = \deg(q_i(x)) + \deg(v_{i-1}(x))$$

Thus

$$\deg(r_{i-1}(x)) = \deg(a(x)) - \deg(v_{i-1}(x)) - \deg(q_i(x)) = \deg(a(x)) - \deg(v_i(x))$$

and so

$$\deg(v_i(x)) + \deg(r_{i-1}(x)) = \deg(a(x))$$

□

7.6 Decoding BCH-Codes Using The Key Equation

In this section, we explain a second algorithm to decode BCH-codes. This new algorithm is very efficient in finding the error-locating polynomial and in finding the error sizes.

We use the same notations as in Section 7.4. In particular, we are assuming that at most t errors occurred during transmission.

We define the *syndrome polynomial* $S(x)$ as

$$S(x) = 1 + S_1x + S_2x^2 + \cdots + S_{2t}x^{2t} = 1 + \sum_{j=1}^{2t} S_jx^j$$

and the *error-evaluating polynomial* $\omega(x)$ as

$$\omega(x) = s(x) + \sum_{i=1}^k X_i^b Y_i x \left(\prod_{j=1, j \neq i}^k (1 - X_j x) \right)$$

Note that for $a \neq 0$, we have that

$$(1 - ax)((ax) + (ax)^2 + \cdots + (ax)^{2t}) \equiv ax - (ax)^{2t+1} \equiv ax \pmod{x^{2t+1}}$$

and so

$$\frac{ax}{1 - ax} \equiv (ax) + (ax)^2 + \cdots + (ax)^{2t} \equiv \sum_{j=1}^{2t} (ax)^j \pmod{x^{2t+1}}$$

Hence we get that

$$\begin{aligned} S(x) - 1 &\equiv \sum_{j=1}^{2t} S_j x^j \pmod{x^{2t+1}} \\ &\equiv \sum_{j=1}^{2t} \left(\sum_{i=1}^k Y_i X_i^{b+j-1} \right) x^j \pmod{x^{2t+1}} \\ &\equiv \sum_{i=1}^k Y_i X_i^{b-1} \left(\sum_{j=1}^{2t} (X_i x)^j \right) \pmod{x^{2t+1}} \\ &\equiv \sum_{i=1}^k Y_i X_i^{b-1} \frac{X_i x}{1 - X_i x} \pmod{x^{2t+1}} \\ &\equiv \sum_{i=1}^k \frac{Y_i X_i^b x}{1 - X_i x} \pmod{x^{2t+1}} \end{aligned}$$

So

$$\begin{aligned}
(S(x) - 1)s(x) &\equiv \sum_{i=1}^k \frac{Y_i X_i^b x}{1 - X_i x} \left(\prod_{j=1}^k (1 - X_j x) \right) \pmod{x^{2t+1}} \\
&\equiv \sum_{i=1}^k Y_i X_i^b x \left(\prod_{j=1, j \neq i}^k (1 - X_j x) \right) \pmod{x^{2t+1}} \\
&\equiv \omega(x) - s(x) \pmod{x^{2t+1}}
\end{aligned}$$

Hence we have the *Key Equation*

$$\boxed{S(x)s(x) \equiv \omega(x) \pmod{x^{2t+1}}}$$

Assume for a moment that we can solve this equation for the error-locating polynomial $s(x)$ and the error-evaluating polynomial $\omega(x)$. Then by finding the roots of $s(x)$, we know the values of X_1, X_2, \dots, X_k . Note that for $l = 1, 2, \dots, k$, we have that

$$\begin{aligned}
\omega(X_l^{-1}) &= s(X_l^{-1}) + \sum_{i=1}^k X_i^b Y_i X_l^{-1} \left(\prod_{j=1, j \neq i}^k (1 - X_j X_l^{-1}) \right) \\
&= X_l^b Y_l X_l^{-1} \left(\prod_{j=1, j \neq l}^k (1 - X_j X_l^{-1}) \right) \\
&= X_l^{b-k} Y_l \left(\prod_{j=1, j \neq l}^k (X_l - X_j) \right)
\end{aligned}$$

So $\omega(X_l^{-1}) \neq 0$ and

$$Y_l = \frac{X_l^{k-b} \omega(X_l^{-1})}{\prod_{j=1, j \neq l}^k (X_l - X_j)}$$

Hence we can find the error sizes quite easily by evaluating $\omega(X_l^{-1})$ for $l = 1, 2, \dots, k$.

Besides being a solution of the Key Equation, $s(x)$ and $\omega(x)$ have some other properties:

1. $\deg(s(x)) = k \leq t$ and $\deg(\omega(x)) \leq k \leq t$
2. $s(0) = 1$
3. $\gcd(s(x), \omega(x)) = 1$

Indeed, since $s(x)$ factors into linear factors and none of the zeroes of $s(x)$ are zeroes of $\omega(x)$ (namely $\omega(X_l^{-1}) \neq 0$ for $l = 1, 2, \dots, k$), we must have that $\gcd(s(x), \omega(x)) = 1$.

Note that given the syndrome polynomial $S(x)$, there are plenty of polynomials $s_1(x), \omega_1(x)$ with $S(x)s_1(x) \equiv \omega_1(x) \pmod{x^{2t+1}}$. However, if $s_1(x)$ and $\omega_1(x)$ satisfy some additional conditions then they have to be $s(x)$ and $\omega(x)$. The next lemma is a step in the right direction.

Lemma 7.11 *Let $S(x)$, $s(x)$ and $\omega(x)$ be as defined before. Suppose that $s_1(x)$ and $\omega_1(x)$ are polynomials such that $S(x)s_1(x) \equiv \omega_1(x) \pmod{x^{2t+1}}$ and $\deg(s_1(x)), \deg(\omega_1(x)) \leq t$. Then there exists a polynomial $\mu(x)$ such that $s_1(x) = \mu(x)s(x)$ and $\omega_1(x) = \mu(x)\omega(x)$.*

Proof : Since $s(x)$ and $\omega(x)$ satisfy the Key Equation, we have that

$$S(x)s(x) \equiv \omega(x) \pmod{x^{2t+1}}$$

Hence

$$s_1(x)S(x)s(x) \equiv s_1(x)\omega(x) \pmod{x^{2t+1}}$$

Similarly, since $s_1(x)$ and $\omega_1(x)$ satisfy the Key Equation, we have that

$$S(x)s_1(x) \equiv \omega_1(x) \pmod{x^{2t+1}}$$

and so

$$s(x)S(x)s_1(x) \equiv s(x)\omega_1(x) \pmod{x^{2t+1}}$$

Hence

$$s_1(x)\omega(x) \equiv s(x)\omega_1(x) \pmod{x^{2t+1}}$$

But $s(x), \omega(x), s_1(x)$ and $\omega_1(x)$ are all polynomials of degree at most t . So

$$\deg(s_1(x)\omega(x)) \leq 2t < 2t + 1 \quad \text{and} \quad \deg(s(x)\omega_1(x)) \leq 2t < 2t + 1$$

Since $s_1(x)\omega(x) \equiv s(x)\omega_1(x) \pmod{x^{2t+1}}$, we must have that

$$s_1(x)\omega(x) = s(x)\omega_1(x)$$

So $s(x)$ divides $s_1(x)\omega(x)$. Since $\gcd(s(x), \omega(x)) = 1$, it follows that $s(x)$ divides $s_1(x)$. So $s_1(x) = \mu(x)s(x)$ for some polynomial $\mu(x)$. Hence

$$s(x)\omega_1(x) = s_1(x)\omega(x) = \mu(x)s(x)\omega(x)$$

and so

$$s(x)\omega_1(x) = \mu(x)s(x)\omega(x)$$

Since $s(0) = 1$, we have that $s(x) \neq 0$. Thus we can divide by $s(x)$. So $\omega_1(x) = \mu(x)\omega(x)$. \square

Using the Euclidean Algorithm, we can now solve the Key Equation for $s(x)$ and $\omega(x)$.

Proposition 7.12 *Let $S(x)$, $s(x)$ and $\omega(x)$ be as defined above. Assume we use the Euclidean Algorithm (see the previous section) with $a(x) = x^{2t+1}$ and $b(x) = S(x)$. At some step j , we get that $\deg(r_j(x)) \leq t$. Then*

$$s(x) = \frac{1}{v_j(0)} v_j(x) \quad \text{and} \quad \omega(x) = \frac{1}{v_j(0)} r_j(x)$$

Proof : From Proposition 7.10(a), we get that

$$r_j(x) = u_j(x)x^{2t+1} + v_j(x)S(x) \tag{*}$$

and so

$$S(x)v_j(x) \equiv r_j(x) \pmod{x^{2t+1}}$$

Note that $\deg(r_j(x)) \leq t$ by assumption. Since step j is the first time that $\deg(r_j(x)) \leq t$, we must have that $\deg(r_{j-1}(x)) > t$. Hence it follows from Proposition 7.10(g) that

$$\deg(v_j(x)) = 2t + 1 - \deg(r_{j-1}(x)) < 2t + 1 - t = t + 1$$

and so $\deg(v_j(x)) \leq t$. By Lemma 7.11, there exists a polynomial $\mu(x)$ such that

$$v_j(x) = \mu(x)s(x) \quad \text{and} \quad r_j(x) = \mu(x)\omega(x)$$

Substituting this into (*), we find that

$$\mu(x)\omega(x) = u_j(x)x^{2t+1} + \mu(x)s(x)S(x)$$

Hence

$$\mu(x)(\omega(x) - s(x)S(x)) = u_j(x)x^{2t+1} \tag{**}$$

Since

$$S(x)s(x) \equiv \omega(x) \pmod{x^{2t+1}}$$

we have that x^{2t+1} divides $\omega(x) - S(x)s(x)$ and so $\omega(x) - S(x)s(x) = x^{2t+1}t(x)$ for some polynomial $t(x)$. Substituting this into (**) gives us that

$$\mu(x)x^{2t+1}t(x) = u_j(x)x^{2t+1}$$

and so

$$\mu(x)t(x) = u_j(x)$$

Since $v_0(x) = 1$ and $\deg(v_i(x)) \geq \deg(v_{i-1}(x))$ for all $i \geq 0$ by Proposition 7.10(f), we get that $v_i(x) \not\equiv 0$ for all $i \geq 0$. So it follows from $v_j(x) = \mu(x)s(x)$ that $\mu(x) \not\equiv 0$. But $u_j(x) = \mu(x)t(x)$ and $v_j(x) = \mu(x)s(x)$. So $\mu(x)$ is a common divisor of $u_j(x)$ and $v_j(x)$. Since $\gcd(u_j(x), v_j(x)) = 1$ by Proposition 7.10(c), we conclude that $\mu(x)$ is a constant polynomial. Since $s(0) = 1$ and $v_j(x) = \mu(x)s(x)$, we find that $v_j(0) = \mu(0)s(0) = \mu(0)$. Hence

$$\mu(x) = v_j(0)$$

So

$$s(x) = \frac{1}{v_j(0)} v_j(x) \quad \text{and} \quad \omega(x) = \frac{1}{v_j(0)} r_j(x) \quad \square$$

Now we can present a second decoding algorithm for BCH-codes. Make sure to understand that we will decode correctly if at most t errors occur during transmission.

1. Upon receiving the word \mathbf{y} , evaluate the syndromes $S_1, S_2, \dots, S_{\delta-1}$ (recall that $S_j = \mathbf{y}(\beta^{b+j-1})$ for all j). If $S_1 = S_2 = \dots = S_{\delta-1} = 0$ then \mathbf{y} is a codeword and we decode as \mathbf{y} ; otherwise proceed to step 2.
2. Put $S(x) = 1 + S_1x + S_2x^2 + \dots + S_{\delta-1}x^{\delta-1}$. Go through the implementation of the Euclidean Algorithm with $a(x) = x^{2t+1}$ and $b(x) = S(x)$ until at step j we have that $\deg(r_j(x)) \leq t$. Put

$$s(x) = \frac{1}{v_j(0)} v_j(x) \quad \text{and} \quad \omega(x) = \frac{1}{v_j(0)} r_j(x)$$

3. Find $0 \leq i_1 < i_2 < \dots < i_k \leq n-1$ such that $X_1 = \beta^{i_1}, X_2 = \beta^{i_2}, \dots, X_k = \beta^{i_k}$ are the reciprocals of the zeros of $s(x)$.
4. Calculate the error sizes :

$$Y_i = \frac{X_i^{k-b} \omega(X_i^{-1})}{\prod_{j=1, j \neq i}^k (X_i - X_j)} \quad \text{for } i = 1, 2, \dots, k$$

5. Put

$$\mathbf{e} = (0, \dots, 0, \underbrace{Y_1}_{\text{position } i_1}, 0, \dots, 0, \underbrace{Y_2}_{\text{position } i_2}, 0, \dots, 0, \underbrace{Y_k}_{\text{position } i_k}, 0, \dots, 0)$$

and decode as $\mathbf{c} := \mathbf{y} - \mathbf{e}$.

We illustrate this decoding algorithm with the same examples as we used when illustrating the PGZ-decoding algorithm on page 70.

Examples

1. A binary BCH-code of length $n = 21$ and designed distance $\delta = 7$.

Let α be a primitive element of $GF(64)$ as given in our table. Put $\beta = \alpha^3$. Then β is an element of order 21. Finally, put $\delta = 7$ and $b = 1$. Then our code C has generator polynomial

$$g(x) = \text{lcm}(m_\beta(x), m_{\beta^2}(x), m_{\beta^3}(x), m_{\beta^4}(x), m_{\beta^5}(x), m_{\beta^6}(x))$$

where $m_{\beta^i}(x)$ is the minimal polynomial of β^i over $GF(2)$.

Suppose we receive the word

$$\mathbf{y} = 001001000000000000000000$$

So $\mathbf{y}(x) = x^2 + x^5$.

First, we calculate $\delta - 1 = 6$ syndromes. The results were

$$S_1 = \alpha^{51}, S_2 = \alpha^{39}, S_3 = \alpha^{36}, S_4 = \alpha^{15}, S_5 = \alpha^{39} \text{ and } S_6 = \alpha^9$$

So the syndrome polynomial is

$$S(x) = 1 + \alpha^{51}x + \alpha^{39}x^2 + \alpha^{36}x^3 + \alpha^{15}x^4 + \alpha^{39}x^5 + \alpha^9x^6$$

Next, we go through the Euclidean Algorithm. We get

$$\begin{array}{rcl} 1 & 0 & x^7 \\ 0 & 1 & \alpha^9x^6 + \alpha^{39}x^5 + \alpha^{15}x^4 + \alpha^{36}x^3 + \alpha^{39}x^2 + \alpha^{51}x + 1 \quad (1) \\ 1 & \alpha^{54}x + \alpha^{21} & \alpha^{42}x^5 + \alpha^9x^4 + \alpha^{48}x^3 + \alpha^6x^2 + \alpha^{18}x + \alpha^{21} \quad (2) \\ \alpha^{30}x & \alpha^{21}x^2 + \alpha^{51}x + 1 & \alpha^{21}x^2 + 1 \quad (3) \end{array}$$

because

$$\begin{aligned} (1) \quad x^7 &= (\alpha^9x^6 + \alpha^{39}x^5 + \alpha^{15}x^4 + \alpha^{36}x^3 + \alpha^{39}x^2 + \alpha^{51}x + 1)(\alpha^{54}x + \alpha^{21}) + \\ &\quad (\alpha^{42}x^5 + \alpha^9x^4 + \alpha^{48}x^3 + \alpha^6x^2 + \alpha^{18}x + \alpha^{21}) \\ \text{so } R_3 &= R_1 - (\alpha^{54}x + \alpha^{21})R_2 \\ (2) \quad \alpha^9x^6 + \alpha^{39}x^5 + \alpha^{15}x^4 + \alpha^{36}x^3 + \alpha^{39}x^2 + \alpha^{51}x + 1 &= \\ &\quad (\alpha^{42}x^5 + \alpha^9x^4 + \alpha^{48}x^3 + \alpha^6x^2 + \alpha^{18}x + \alpha^{21})\alpha^{30}x + (\alpha^{21}x^2 + 1) \\ \text{so } R_4 &= R_2 - \alpha^{30}xR_3 \\ (3) \quad \deg(\alpha^{21}x^2 + 1) &\leq 3 = t \text{ so we stop} \end{aligned}$$

Hence

$$v_2(x) = \alpha^{21}x^2 + \alpha^{51}x + 1 \quad \text{and} \quad r_2(x) = \alpha^{21}x^2 + 1$$

Since $v_2(0) = 1$, we can calculate the error-locating polynomial $s(x)$ and the error-evaluating polynomial $\omega(x)$:

$$s(x) = \alpha^{21}x^2 + \alpha^{51}x + 1 \quad \text{and} \quad \omega(x) = \alpha^{21}x^2 + 1$$

Next, we need to find the roots of $s(x)$. We should find two roots and they should be powers of β . The results were that the roots are $\beta^{-2} = \alpha^{-6}$ and $\beta^{-5} = \alpha^{-15}$. Hence

$$X_1 = \beta^2 = \alpha^6 \quad \text{and} \quad X_2 = \beta^5 = \alpha^{15}$$

That means that the first error occurred in the second position while the second error occurred in the fifth position (recall that we start numbering the position from zero).

Finally, we calculate the error sizes Y_1 and Y_2 . In this case (namely $b = 1$ and $k = 2$), we get that

$$Y_1 = \frac{X_1\omega(X_1^{-1})}{X_1 - X_2} \quad \text{and} \quad Y_2 = \frac{X_2\omega(X_2^{-1})}{X_2 - X_1}$$

We get that

$$\begin{aligned}\omega(X_1^{-1}) &= \omega(\alpha^{-6}) = \alpha^{21}\alpha^{-12} + 1 = \alpha^9 + 1 = 011000 + 000001 = 011001 = \alpha^{45} \\ \omega(X_2^{-1}) &= \omega(\alpha^{-15}) = \alpha^{21}\alpha^{-30} + 1 = \alpha^{-9} + 1 = \alpha^{54} + 1 = 010111 + 000001 = 010110 = \alpha^{36} \\ X_2 - X_1 &= \alpha^{15} + \alpha^6 = 101000 + 000011 = 101011 = \alpha^{51}\end{aligned}$$

Hence

$$Y_1 = \frac{\alpha^6\alpha^{45}}{\alpha^{51}} = \frac{\alpha^{51}}{\alpha^{51}} = 1 \quad \text{and} \quad Y_2 = \frac{\alpha^{15}\alpha^{36}}{\alpha^{51}} = \frac{\alpha^{51}}{\alpha^{51}} = 1$$

So the error vector is

$$\mathbf{e} = 001001000000000000000000000000$$

Hence we decode \mathbf{y} as

$$\mathbf{y} - \mathbf{e} = \mathbf{0} = 000000000000000000000000000000$$

2. A BCH-code over $GF(4)$ of length $n = 5$ and designed distance $\delta = 3$.

Let α be a primitive element of $GF(16)$ as given in our table. Then $GF(4) = \{0, 1, \alpha^5, \alpha^{10}\}$ is a subfield of $GF(16)$. Put $\beta = \alpha^3$. Then β is an element of order 5. Finally, put $\delta = 3$ and $b = 2$. Then our code C has generator polynomial

$$g(x) = \text{lcm}(m_{\beta^2}(x), m_{\beta^3}(x))$$

where $m_{\beta^i}(x)$ is the minimal polynomial of β^i over $GF(4)$.

Suppose we receive

$$\mathbf{y} = 110\alpha^{10}1$$

So $\mathbf{y}(x) = 1 + x + \alpha^{10}x^3 + x^4$.

First we calculate the two syndromes. The results were

$$S_1 = \alpha^{11} \quad \text{and} \quad S_2 = \alpha^{14}$$

So the syndrome polynomial is

$$S(x) = 1 + \alpha^{11}x + \alpha^{14}x^2$$

Next, we go through the Euclidean Algorithm. We get

$$\begin{array}{rcl} 1 & 0 & x^3 \\ 0 & 1 & \alpha^{14}x^2 + \alpha^{11}x + 1 \quad (1) \\ 1 & \alpha x + \alpha^{13} & \alpha^7x + \alpha^{13} \quad (2) \end{array}$$

because

$$\begin{aligned}(1) \quad x^3 &= (\alpha^{14}x^2 + \alpha^{11}x + 1)(\alpha x + \alpha^{13}) + (\alpha^7x + \alpha^{13}) \quad \text{so } R_3 = R_1 - (\alpha x + \alpha^{13})R_2 \\ (2) \quad \deg(\alpha^7x + \alpha^{13}) &\leq 1 = t \quad \text{so we stop}\end{aligned}$$

Hence

$$v_1(x) = \alpha x + \alpha^{13} \quad \text{and} \quad r_1(x) = \alpha^7 x + \alpha^{13}$$

Since $v_1(0) = \alpha^{13}$, we get that

$$s(x) = \frac{1}{\alpha^{13}} (\alpha x + \alpha^{13}) = \alpha^2(\alpha x + \alpha^{13}) = \alpha^3 x + 1$$

and

$$\omega(x) = \frac{1}{\alpha^{13}} (\alpha^7 x + \alpha^{13}) = \alpha^2(\alpha^7 x + \alpha^{13}) = \alpha^9 x + 1$$

Next, we need to find the roots of $s(x)$. We found that

$$x = \frac{1}{\alpha^3} = \alpha^{-3}$$

So

$$X_1 = \alpha^3 = \beta^1$$

That means that the error occurred in the first position (recall that we start numbering the position from zero).

Finally, we find the error size Y_1 . In this case ($b = 2$ and $k = 1$), we have

$$Y_1 = X_1^{-1} \omega(X_1^{-1}) = \alpha^{-3}(\alpha^9 \alpha^{-3} + 1) = \alpha^{12}(\alpha^6 + 1) = \alpha^{18} + \alpha^{12} = \alpha^3 + \alpha^{12} = 1000 + 0011 = 1011 = \alpha^5$$

So the error vector is

$$\mathbf{e} = 0\alpha^5 000$$

Hence we decode \mathbf{y} as

$$\mathbf{y} - \mathbf{e} = 110\alpha^{10} 1 - 0\alpha^5 000 = 1\alpha^{10} 0\alpha^{10} 1$$

since $1 + \alpha^5 = 0001 + 1011 = 1010 = \alpha^{10}$. ▷