An-Najah National University

Faculty of Graduate Studies

Error-Detecting and Error-Correcting Using Hamming and Cyclic Codes

By

Ne'am Hashem Ibraheem Ibraheem

Supervisor:

Dr. "Mohammad Othman" Omran

This Thesis is Submitted in Partial Fulfillment of the Requirements for the Degree of Master of Science in Mathematics, Faculty of Graduate Studies, An-Najah National University, Nablus, Palestine.

2009

Error – Detecting and Error-Correcting Using Hamming and Cyclic Codes

By
Ne'am Hashem Ibraheem Ibraheem

This thesis was defended successfully on 6/12/2009 and approved by

Committee Members		**Signature**
1- Dr. "Mohammad Othman" Omran	Supervisor	
2- Dr. Anwar Saleh	Internal Examiner	
3- Dr. Ali Abed Almohesen	External Examiner	

Dedication

To my parents "Hashem & Amal"

Acknowledgments

I would like to thank Dr. Mohammad Omran for his guidance and suggestions, as well as work's associates in Hisham Hijjawi college of technology.

إقــرار

أنا الموقع أدناه، مقدم الرسالة التي تحمل العنوان :

Error-Detecting and Error-Correcting Using Hamming and Cyclic Codes

اكتشاف الأخطاء وتصحيحها باستخدام شيفرات هامنج والشيفرات الحلقية

أقر بأن ما اشتملت عليه هذه الرسالة انما هي نتاج جهدي الخاص، باستثناء ما تمت الاشارة اليه حيثما ورد، وان هذه الرسالة ككل، أو أي جزء منها لم يقدم من قبل لنيل أي درجة أو لقب علمي أو بحثي لدى أي مؤسسة تعليمية أو بحثية أخرى .

Declaration

The work provided in this thesis, unless otherwise referenced, is the researcher's own work, and has not been submitted elsewhere for any other degree or qualification.

Student's Name اسم الطالب:

Signature التوقيع:

Date: التاريخ

List of contents

List of Tables

List of figures

Error-Detecting and Error-Correcting Using Hamming and Cyclic codes

By:
Ne'am Hashem Ibraheem Ibraheem
Supervisor by :
Dr. "Mohammad Othman" Omran

Abstract

In this thesis we provide an overview of two types of linear block codes: Hamming and cyclic codes. We study the generation, encoding and decoding of these codes as well as studying schemes and/or algorithms of error-detecting and error-correcting of these codes.

Preface

Coding theory is concerned with the transmission of data across noisy channels and the recovery of corrupted messages. It has found widespread applications in electrical engineering, digital communication, mathematics and computer science. While the problems in coding theory often arise from engineering applications, it is fascinating to note the crucial role played by mathematics in the development of the field.

The importance of algebra in coding theory is a commonly acknowledged fact, with many deep mathematical results being used in elegant ways in the advancement of coding theory; therefore coding theory appeals not just to engineers and computer scientists, but also to mathematicians and hence, coding theory is sometimes called *algebraic coding theory*.

An algebraic techniques involving finite fields, group theory, polynomial algebra as well as linear algebra deal with the design of error-correcting codes for the reliable transmission of information across noisy channels.

Usually, coding is divided into two parts:

1. Source coding:
   - Source encoding
   - Source decoding
2. Channel coding:
   - Channel encoding
   - Channel decoding

Source encoding involves changing the message source to a suitable code say u to be transmitted through the channel. Channel encoding deals with

the source encoded message u, by introducing some extra data bits that will be used in detecting and/or even correcting the transmitted message. Thus the result of the source encoding is a codeword, say v. Likewise, channel decoding and source decoding are applied on the destination side to decode the received codeword r as correctly as possible. Figure 1 represents a model of a data transmission system.

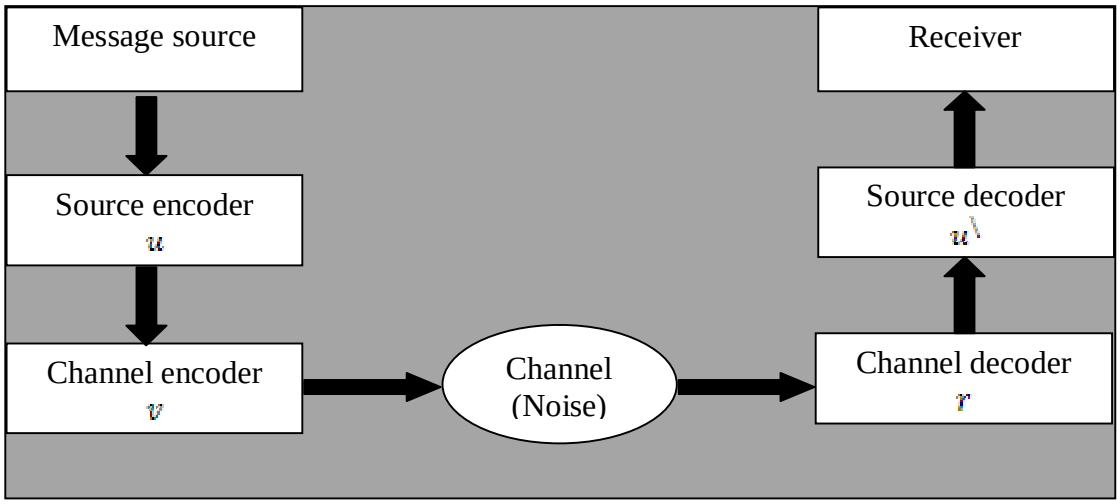


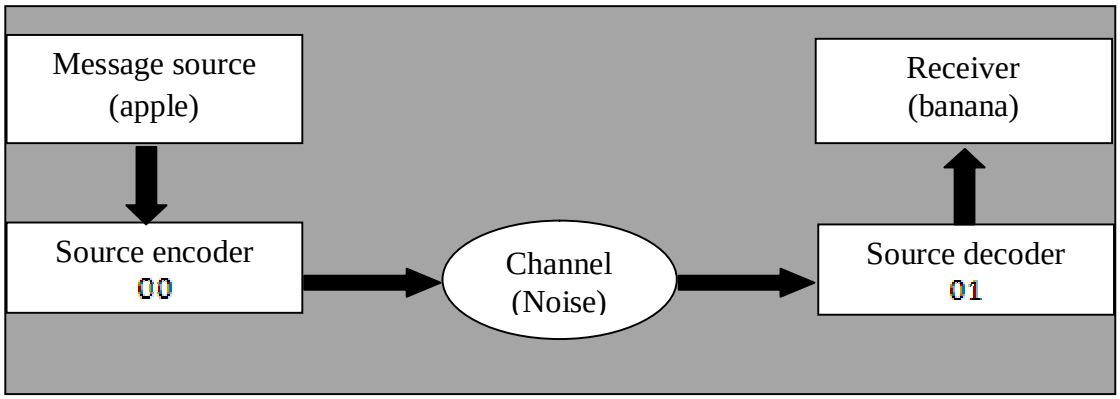
Figure 1: Model of a Data Transmission System

For example: Consider a message source of four fruit words to be transmitted: apple, banana, cherry and grape. The source encoder encodes these words into the following binary data ($u_1\ u_2\ u_3\ u_4$):

apple$\rightarrow u_1 = (0\ 0)$, banana$\rightarrow u_2 = (0\ 1)$, cherry$\rightarrow u_3 = (1\ 0)$, grape$\rightarrow u_4 = (1\ 1)$.

Suppose the message 'apple' is to be transmitted over a noisy channel. The bits $u_1 = (0\ 0)$ will be transmitted instead. Suppose an error of one bit occurred during the transmission and the code $(0\ 1)$ is received instead as seen in the following figure. The receiver may not realize that the message

was corrupted and the received message will be decoded into 'banana'. These a communication error occurred.



With channel coding, this error may be detected (and even corrected) by introducing a redundancy bit as follows $(v_1\ v_2\ v_3\ v_4)$:

$$(0\ 0) \rightarrow v_1 = (0\ 0\ 0), (0\ 1) \rightarrow v_2 = (0\ 1\ 1),$$
$$(1\ 0) \rightarrow v_3 = (1\ 0\ 1), (1\ 1) \rightarrow v_4 = (1\ 1\ 0)$$

The newly encoded message 'apple' is now $(0\ 0\ 0)$. Suppose this message was transmitted and an error of one bit only occurred. The receiver may get one of the following: $(1\ 0\ 0), (0\ 1\ 0)$ or $(0\ 0\ 1)$. In this way, we can detect the error, as none of $(1\ 0\ 0), (0\ 1\ 0)$ or $(0\ 0\ 1)$ is among our encoded messages.

Note that the above channel encoding scheme does not allow us to correct errors. For instance, if $(1\ 0\ 0)$ is received, then we do not know whether $(1\ 0\ 0)$ comes from $(0\ 0\ 0)$, $(1\ 0\ 1)$ or $(1\ 1\ 0)$. However, if more three redundancy bits are introduced instead of one bit, we will be able to correct errors. For instance, we can design the following channel coding scheme:

$$(0\ 0) \rightarrow (0\ 0\ 0\ 0\ 0), (0\ 1) \rightarrow (0\ 1\ 1\ 1\ 1),$$

$$(1\ 0) \rightarrow (1\ 0\ 1\ 1\ 0), (1\ 1) \rightarrow (1\ 1\ 0\ 0\ 1)$$

Again if the message $(0\ 0\ 0\ 0\ 0)$ was transmitted over a noisy channel and that there is only one error introduced, then the received word must be one of the following five: $(1\ 0\ 0\ 0\ 0), (0\ 1\ 0\ 0\ 0), (0\ 0\ 1\ 0\ 0), (0\ 0\ 0\ 1\ 0)$ or $(0\ 0\ 0\ 0\ 1)$. Since only one error occurred and since each of these five codes differs from $(0\ 0\ 0\ 0\ 0)$ by only one bit, and from the other three correct codes $(0\ 1\ 1\ 1\ 1), (1\ 0\ 1\ 1\ 0)$ and $(1\ 1\ 0\ 0\ 1)$ by at least two bits, then the receiver will decode the received message into $(0\ 0\ 0\ 0\ 0)$ and, hence, the received message will be correctly decoded into 'apple'.

Algebraic coding theory is basically divided into two major types of codes: Linear block codes and Convolutional codes.

In this thesis we present some encoding and decoding schemes as well as some used error detection/correction coding techniques using linear block codes only. We discuss only two types of linear block codes: Hamming and cyclic codes.

History

The history of data-transmission codes began in 1948 with the publication of a famous paper by Claude Shannon. Shannon showed that associated with any communication channel or storage channel is a number *C* (measured in bits per second), called the capacity of the channel, which has the following significance: Whenever the information transmission rate *R* (in bits per second) required of a communication or storage system is less than C then, by using a data-transmission code, it is possible to design a communication system for the channel whose probability of output error is as small as desired. Shannon, however, did not tell us how to find suitable codes; his contribution was to prove that they exist and to define their role.

Throughout the 1950s, much effort was devoted to finding explicit constructions for classes of codes. The first block codes were introduced in 1950 when Hamming described a class of single-error-correcting block codes and he published what is now known as Hamming code, which remains in use in many applications today.

In 1957, Among the first codes used practically were the cyclic codes which were generated using shift registers. It was quickly noticed by Prange that the cyclic codes have a rich algebraic structure, the first indication that algebra would be a valuable tool in code design.

In the 1960s, the major advances came in 1960 when Hocquenghem and Bose and Ray-Chaudhuri found a large class of multiple-error-correcting codes (the BCH codes). The discovery of BCH codes led to a search for practical methods of designing the hardware or software to implement the

encoder and decoder. In the same year independently, Reed, Solomon and Arimoto found a related class of codes for nonbinary channels. Concatenated codes were introduced by Forney (1966), later Justesen used the idea of a concatenated code to devise a completely constructive class of long block codes with good performance.

During the 1970s, these two avenues of research began to draw together in some ways and to diverge further in others. Meanwhile, Goppa (1970) defined a class of codes that is sure to contain good codes, though without saying how to identify the good ones.

The 1980s saw encoders and decoders appear frequently in newly designed digital communication systems and digital storage systems.

The 1990s witnesses an evaluation of all groups in informatics at the universities in Norway. The evaluation was performed by a group of internationally recognized experts. The committee observed that the period 1988-92, had the largest number of papers (27) published in internationally refereed journals among all the informatics groups in Norway. In the period 1995-1997 the goal of finding explicit codes which reach the limits predicted by Shannon's original work has been achieved. The constructions require techniques from a surprisingly wide range of pure mathematics: linear algebra, the theory of fields and algebraic geometry all play a vital role. Not only has coding theory helped to solve problems of vital importance in the world outside mathematics, it also has enriched other

branches of mathematics, with new problems as well as new solutions. In 1998 Alamouti described a space-time code.

In 2000 Aji, McEliece and others synthesize several decoding algorithms using message passing ideas. In the period 2002-2006 many books and papers are introduce such as Algebraic soft-Decision Decoding of Reed-Solomon Codes by Koetter R., and Error Control Coding: Fundamentals and Applications by Lin and Costello and Error Correction Coding by Moon T. in 2005.

During This decade, development of algorithms for hard-decision decoding of large nonbinary block codes defined on algebraic curves. Decoders for the codes known as hermitian codes are now available and these codes may soon appear in commercial products. At the same time, the roots of the subject are growing even deeper into the rich soil of mathematics.

Chapter 1

Introduction to Algebra

As mentioned earlier in the Preface, the study of linear block codes requires basic knowledge in modern algebra and linear algebra. Hence, in this chapter we provide the reader with basic definitions and terminologies that help in the understanding of the material in this thesis. Groups, fields, vector spaces and other definitions and concepts in algebra that relate to linear block codes are discussed in this chapter.

1.1 Groups

Definition 1.1: Let G be a set of elements. A binary operation $*$ on G is a rule that assigns to each pair a and b of G a unique element $c = a * b$ in G. We say that G is closed under $*$.

Definition 1.2: A group $(G,*)$ (or simply G) is a set G of elements together with a binary operation $*$ on G such that:

(i) The operation $*$ is associative:

For any $a, b, c \in G$, $(a * b) * c = a * (b * c)$.

(ii) There is a unique element $e \in G$, called the identity element, such that

$a * e = e * a = a \ \forall a \in G$.

(iii) For every $a \in G$, there is a unique element $a^{-1} \in G$, called the inverse of a, such that $a * a^{-1} = a^{-1} * a = e$.

Theorem 1.1: A group G is said to be commutative if its binary operation $*$ is commutative, i.e., for every $a, b \in G$, $a * b = b * a$.

Definition 1.3: The order of a group G, denoted by $|G|$, is the number of elements in G if G has a finite number of elements, and is ∞ otherwise. A group G is finite (infinite) if $|G|$ is finite (infinite).

Example 1.1: Consider the set $G = Z_2 = \{0, 1\}$. Let the binary operation $*$ be denoted by $\oplus$ and defined as follows:

Table 1: modulo-2 addition

$\oplus$	0	1
0	0	1
1	1	0

This binary operation $\oplus$ is called modulo-2 addition. Is easy to check that G is a communication group under $\oplus$.

For simplicity the modulo-2 addition operation $\oplus$ will be denoted by $+$.

1.2 Permutation Groups

Definition 1.4: A permutation σ of a nonempty set $A = \{1, 2, \ldots, n\}$ is a one-to-one mapping of the set A onto itself. This permutation σ can be denoted by:

$$\sigma = \begin{pmatrix} 1 & \ldots & n \\ \sigma(1) & \ldots & \sigma(n) \end{pmatrix}, \text{ where } \sigma(i) = j \in A \ \ \forall\, i, j = 1, 2, \ldots, n.$$

Note that, the order of the columns in this representation of σ is immaterial. For example, $\begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix} = \begin{pmatrix} 2 & 3 & 1 \\ 3 & 1 & 2 \end{pmatrix}$.

The set of all permutations on $A = \{1, 2, \ldots, n\}$ is denoted by S_n.

The composition operation, denoted by $\circ$, on S_n is defined by $\sigma_2 \circ \sigma_1$, where σ_1 is applied first and then σ_2, for any two permutations $\sigma_1, \sigma_2 \in S_n$.

Clearly $\sigma_2 \circ \sigma_1$ is again a permutation in S_n, so $S_n$is closed under the operation o. $1 = \begin{pmatrix} 1 & 2 & \dots & n \\ 1 & 2 & \dots & n \end{pmatrix}$, is the identity permutation in S_n.

The inverse of any $\sigma = \begin{pmatrix} 1 & \dots & n \\ \sigma(1) & \dots & \sigma(n) \end{pmatrix}$ is $\sigma^{-1} = \begin{pmatrix} \sigma(1) & \dots & \sigma(n) \\ 1 & \dots & n \end{pmatrix}$, which is itself a permutation in S_n.

Composition of permutations is associative; for σ_1, σ_2 and σ_3, we have $(\sigma_1 \circ \sigma_2) \circ \sigma_3 = \sigma_1 \circ (\sigma_2 \circ \sigma_3)$. Thus we have this theorem:

Theorem 1.2: The set $(S_n, \circ)$ is a group.

Note that the composition is not commutative since $\sigma_2 \circ \sigma_1 \neq \sigma_1 \circ \sigma_2$.

So $(S_n, \circ)$ is not a commutative group.

Remark 1.1: The number of elements of $S_n = n!$.

Example 1.2: Let $A = \{1, 2, 3\}$, then the set of all permutations on A is denoted by S_3 and $|S_3| = 3! = 6$ elements, where

$$S_3 = \{1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 2 & 3 \end{pmatrix}, \sigma_1 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \end{pmatrix}, \sigma_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 1 & 2 \end{pmatrix},$$
$$\mu_1 = \begin{pmatrix} 1 & 2 & 3 \\ 1 & 3 & 2 \end{pmatrix}, \mu_2 = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}, \mu_3 = \begin{pmatrix} 1 & 2 & 3 \\ 2 & 1 & 3 \end{pmatrix}\}$$

O	1	σ_1	σ_2	μ_1	μ_2	μ_3
1	1	σ_1	σ_2	μ_1	μ_2	μ_3
σ_1	σ_1	σ_2	1	μ_3	μ_1	μ_2
σ_2	σ_2	1	σ_1	μ_2	μ_3	μ_1
μ_1	μ_1	μ_2	μ_3	1	σ_1	σ_2
μ_2	μ_2	μ_3	μ_1	σ_2	1	σ_1
μ_3	μ_3	μ_1	μ_2	σ_1	σ_2	1

From the above table we have $(S_3, \circ)$ is a group.

Definition 1.5: The pair $(H,*)$ is said to be a subgroup of given group $(G,*)$ if H is a nonempty subset of G and is itself a group under the same operation $*$ of G.

Example 1.3: Let $H = \{1, \sigma_1, \sigma_2\}$. The pair $(H, \circ)$ is a subgroup of S_3 in Example 1.2, according to the following table.

$\circ$	1	σ_1	σ_2
1	1	σ_1	σ_2
σ_1	σ_1	σ_2	1
σ_2	σ_2	1	σ_1

1.3 Cyclic permutations

Definition 1.6: A permutation $\sigma \in S_n$ is called a cycle of length k or k-cyclic $(k \leq n)$ if there exists a list of distinct integers $a_1\ a_2\ \dots\ a_k \in S$ such that

$$\sigma(a_i) = a_{i+1},\ \ i = 1, \dots, k-1$$

$$\sigma(a_k) = a_1$$

$$\sigma(a) = a \ \ if\ a \notin \{a_1\ a_2\ \dots\ a_k\}.$$

In this case σ will be denoted by $(a_1 \dots a_k)$. A cycle of length 2 is called a transposition.

Remark 1.2: The transposition $(a_1\ \ a_2)$ is its own inverse.

Observe that $(a_1\ a_2\ \dots\ a_k), (a_2\ \dots\ a_k a_1), \dots \& (a_k\ a_1\ \dots\ a_{k-1})$ represent the same cycle.

Example 1.4: The permutation $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 3 & 1 & 2 & 5 \end{pmatrix} \in S_5$ is the 4-cycle $(1\ 4\ 2\ 3)$.

Remark 1.3: The composition of disjoint cycles is commutative; i.e, if σ_1, σ_2 are disjoint cycles then $\sigma_1 o \sigma_2 = \sigma_2 o \sigma_1$.

Theorem 1.3: Every permutation in S_n can be expressed uniquely (up to order) as a product of disjoint cycles.

Example 1.5: In S_6, the permutation $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 \\ 5 & 4 & 3 & 2 & 6 & 1 \end{pmatrix}$ can be expressed as a product $(1\ 5\ 6)(2\ 4)$ or $(2\ 4)(1\ 5\ 6)$.

Theorem 1.4: Every cycle of length k is a product of $k-1$ transpositions.

Example 1.6: $\sigma = (1\ 4\ 2\ 3) = (1\ 3)(1\ 2)(1\ 4)$.

1.4 Cyclic Groups & the Order of an Element

Definition 1.7: Let a be an element in a group $(G,*)$, then the set $H = \{ a^n = \underbrace{a * \ldots * a}_{n-times} \mid n \in Z^+ \}$ is called a cyclic subgroup *of* G generated by a, written $H = \langle a \rangle$, and a is called a generator of H.

By convention, $a^0 = 1$.

Definition 1.8: If $G = \langle a \rangle$, then the group $(G,*)$ is said to be cyclic.

Example 1.7: The group $(Z_5, +)$, where $Z_5 = \{0,1,2,3,4\}$, is cyclic. Since every element in Z_5 can be generated by $a = 2$;
$a^1 = 2,\ a^2 = 2+2 = 4,\ a^3 = a^2 + a = 4+2 = 1,\ a^4 = a^3 + a = 1+2 = 3,\ a^5 = a^4 + a = 3+2 = 0$

. So $Z_5 = \langle 2 \rangle$.

Definition 1.9: Let a be an element in a group $(G,*)$, then the smallest positive integer n such that a^n equals to the identity in G is said to be the order of a and is denoted by $ord(a)$. If no such n exists then a is said to be of infinite order.

Remark 1.4: The order of an element should not be confused with the order of a group, which is the number of elements in the group.

Remark 1.5: The order of the k-cycle in S_n is k.

Remark 1.6: If $\sigma \in S_n$ is a product of disjoint cycles say $\sigma = \sigma_1\sigma_2 \ldots \sigma_r$, then the order of σ denoted by $ord(\sigma)$ is the least common multiple(lcm) of the orders of these disjoint cycles. i.e.,
$ord(\sigma) = lcm\{ord(\sigma_1), \ldots, ord(\sigma_r)\}$.

Example 1.8: Let $\sigma = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 3 & 5 & 4 & 1 & 2 & 7 & 6 \end{pmatrix} \in S_7$. Then;

$\sigma = (1\ 3\ 4)(2\ 5)(6\ 7)$, and the order of σ is:

$ord(\sigma) = lcm(3,2,2) = 6$.

1.5 Cosets

Definition 1.10: Let H be a subgroup of a group $(G,*)$ and $a \in G$. The set $a * H = \{a * h : h \in H\}$ is called a left coset of H in G. Similarly, $H * a = \{h * a : h \in H\}$ is called a right coset of H in G.

Note that we can write $aH(Ha)$ instead of $a * H(H * a)$

Of course, in a commutative group, the left and right cosets are the same.

Definition 1.11: Let H be a subgroup of a finite group G. The number of distinct left (right) cosets of H in G denoted by $[G:H] = \frac{|G|}{|H|}$.

Example 1.9: Consider the group $S_3 = \{1, \sigma_1, \sigma_2, \mu_1, \mu_2, \mu_3\}$ as given in Example 1.2 and consider the subgroup $H = \{1, \mu_1\}$.
Then there are $[G:H] = \frac{|G|}{|H|} = \frac{6}{2} = 3$ left cosets as well as 3 right cosets.

The left cosets of H are found as follows:

$1\mathrm{H} = (2\ 3)\mathrm{H} = \mathrm{H} = \{1, \mu_1\}$

$(1\ 2\ 3)H = (1\ 2)H = \{(1\ 2\ 3), (1\ 2)\} = \{\sigma_1, \mu_3\}$

And $(1\ 3\ 2)H = (1\ 3)H = \{(1\ 3\ 2), (1\ 3)\} = \{\sigma_2, \mu_2\}$

Thus the left cosets of H are $\{H, \{(1\ 2\ 3), (1\ 2)\}, \{(1\ 3\ 2), (1\ 3)\}\}$.

The right cosets of H are found in a similar way, and they are $\{\{1, \mu_1\}, \{\sigma_1, \mu_3\}, \{\sigma_2, \mu_2\}\}$.

Theorem 1.5: For a subgroup H of G, left cosets of H satisfiy the following:

(i) If $a \in G$ and $b \in aH$, then $bH = aH$.

(ii) $aH = bH$ if and only if $a^{-1}b \in H$.

(iii) Any two left cosets of H say aH & bH are either equal or disjoint.

(iv) $G = \bigcup_{a \in G} aH$, where the union runs over the set of distinct cosets of H in G.

Fact 1.1: Left cosets of a subgroup H of G define an equivalence relation on G:

(i) Reflexive: $a \in aH$.

(ii) Symmetric: if $a \in bH$ then $b \in aH$.

(iii) Transitive: if $a \in bH$ & $b \in cH$ then $a \in cH$.

Note that theorem 1.5 and fact 1.1 apply for right cosets as well.

1.6 Fields

Definition 1.12: Let F be a nonempty set on which two binary operations, addition " $+$ " and multiplication "." are defined. Then the system $(F, +, .)$ is a field if the following conditions are satisfied:

(i) $(F, +)$ is a commutative group.

(ii) $(F - \{0\}, .)$ is a commutative group.

(iii) Multiplication is distributive over addition; that is, for any three elements a, b *and* c in F: $a.(b + c) = a.b + a.c$

$$(b + c).a = b.a + c.a$$

The elements of the field F are called scalars. The field $(F, +, .)$ will be denoted by F as long as the operations (+) and (.) are understood from the context.

Definition 1.13: If F has a finite number of elements, it is said to be a finite field. The number of elements in the field F is called the order of the field F and is denoted by $|F|$.

Remark 1.7: The set $Z_p = \{0, 1, \ldots, p-1\}$ (where p is prime) is a field of order p under modulo-p addition and multiplication. This field is called a prime field.

Example 1.10: The set $Z_2 = \{0, 1\}$ is a field of order 2 under modulo-2 addition and modulo-2 multiplication. It has the following addition and multiplication tables:

Table 2: modulo-2 addition

+	0	1
0	0	1
1	1	0

Table 3: modulo-2 multiplication

.	0	1
0	0	0
1	0	1

This field is called a binary field and it satisfies: $1 + 1 = 0$, so the addition and subtraction are interchangeable**.**

Basic properties of finite fields:

In the following, let F be a finite field of order p, where p is a prime number.

(i) For every $a \in F, a.0 = 0.a = 0$

(ii) For any two nonzero elements $a, b \in F, a.b \neq 0$.

(iii) $a.b = 0$ and $a \neq 0$ imply that $b = 0$.

(iv) Let $0 \neq a \in Z_p \overset{then}{\Longrightarrow} \underbrace{a.a.....a}_{p-1\ times} = a^{p-1} = 1.$

(v) All finite fields are also called Galois fields and denoted by GF.

According to (v) the prime field Z_p will be denoted by $GF(p)$, hence, the binary field Z_2 in Example 1.10 is denoted by $GF(2)$.

1.7 Polynomials over the Binary Field

A polynomial $f(x)$ of degree n over $GF(2)$ is a polynomial $f(x)$ with coefficients from $GF(2)$, i.e,

$f(x) = f_0 + f_1 x + f_2 x^2 + \cdots + f_n x^n$ where $f_i = 0$ or $\mathbf{1}$ for $0 \leq i \leq n$.

Theorem 1.6: Over $GF(2)$ there are 2^n polynomials of degree n.

Example 1.11: There are four polynomials of degree 2 over $GF(2)$ and they are: $X^2, 1 + X^2$, $X + X^2$ and $1 + X + X^2$.

Now, the polynomials over $GF(2)$ can be added (or subtracted), multiplied and divided modulo 2. Let $g(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_m x^m$ be a polynomial over $GF(2)$ and let $f(x)$ be as above. Then:

$$(f+g)(x) = \sum_{i=0}^{\max\{m,n\}} (f_i + g_i)x^i \text{ where } \begin{cases} g_i = 0 & i > m \\ f_i = 0 & i > n \end{cases}$$

$$\text{and } (f.g)(x) = \sum_{i=0}^{m+n} \left(\sum_{k=0}^{i} f_{i-k} \cdot g_k\right) x^i.$$

When $f(x)$ is divided by $g(x)$, we obtain a unique pair of polynomials: the quotient $q(x)$ and the remainder $r(x)$ over $GF(2)$ with degree of $r(x)$ is less than that of $g(x)$. We then write $f(x) = q(x)g(x) + r(x)$.

Example 1.12: Let $f(x) = 1 + x^2 + x^3 \& g(x) = x + x^2$

$(f + g)(x) = (1 + 0) + (0 + 1)x + (1 + 1)x^2 + (1 + 0)x^3$

Hence, $(f + g)(x) = 1 + x + x^3 = (f - g)(x)$.

$$(f.g)(x) = (1.0) + \big((0.1) + (1.1)\big)x + \big((1.1) + (0.1) + (1.0)\big)x^2 + \big((1.0) + (0.1) + (1.1) + (1.0)\big)x^3 + \big((1.0) + (0.0) + (1.1) + (1.1) + (0.0)\big)x^4 + (1.1)x^5$$
$$= x + x^2 + x^3 + x^5.$$

Example 1.13: Divide $f(x) = x^4 + x^3 + x^2$ by $g(x) = x + 1$ using long-division over $GF(2)$:

$$\begin{array}{r} x^3 + x + 1 \\ x+1 \overline{)\, x^4 + x^3 + x^2} \\ \underline{x^4 + x^3} \\ x^2 \\ \underline{x^2 + x} \\ x \\ \underline{x + 1} \\ 1 \end{array}$$

$q(x) = x^3 + x + 1$ and $r(x) = 1$. We then have

$(x) = x^4 + x^3 + x^2 = (x^3 + x + 1)(x + 1) + 1$

Remark 1.8: If $r(x)$=0 we say that $f(x)$ is divisible by $g(x)$ or $g(x)$ is a factor of $f(x)$.

Theorem 1.7: If a is a root of a polynomial $f(x)$ then $f(x)$ is divisible by $(x - a)$.

Fact 1.2: If we have a polynomial over $GF(2)$ with an even number of terms, then it is divisible by $(x+1)$ because this polynomial has the number 1 as a root.

Example 1.14: Let $f(x) = 1 + x^2 + x^3 + x^4$. Consider $f(1) = 1 + 1 + 1 + 1 = 0 \Rightarrow f(x)$ is divisible by $(x+1)$

Definition 1.14: A polynomial $p(x)$ of degree m over $GF(2)$ is said to be irreducible over $GF(2)$ if $p(x)$ is not divisible by any polynomial over $GF(2)$ of degree less than m but greater than zero. Otherwise $p(x)$ is reducible.

Example 1.15: Let $f(x) = x^3 + x + 1$ be a polynomial over $GF(2)$. $f(x)$ does not have neither "0" nor "1" as roots. So $f(x)$ is not divisible by any polynomial of degree 1: x nor $x+1$. Consequently, it cannot be divisible by a polynomial of degree 2. So $f(x)$ is irreducible over $GF(2)$.

Theorem 1.8: Any irreducible polynomial over $GF(2)$ of degree m divides $x^{(2^m-1)} + 1$.

Example 1.16: You can check that the polynomial $f(x) = x^3 + x + 1$ as in Example 1.15 divides $x^{2^3-1} + 1 = x^7 + 1$ since

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

Definition 1.15: An irreducible polynomial $p(x)$ of degree m is said to be primitive if the smallest positive integer n for which $p(x)$ divides $x^n + 1$ is $n = 2^m - 1$. Otherwise $p(x)$ is not a primitive.

Remark 1.9: In modulo-2 addition we have the following;

$(a + b)^2 = a^2 + b^2$

Theorem 1.9: Let $f(x)$ be a polynomial over $GF(2)$ then for any $i \geq 0$ we have the following:

$$[f(x)]^{2^i} = f(x^{2^i}) \tag{1.1}$$

Proof: Let $f(x) = f_0 + f_1 x + \cdots + f_n x^n$

Then using remark 1.9, we have:

$f^2(x) = (f_0 + f_1 x + \cdots + f_n x^n)^2$

$= f_0^2 + (f_1 x)^2 + (f_2 x^2)^2 + \cdots + (f_n x^n)^2$

Since $f_i = 0$ or 1 $\quad f_i^2 = f_i$ we have:

$= f_0 + f_1 x^2 + f_2 (x^2)^2 + \ldots + f_n (x^2)^n$

$= f(x^2) \Longrightarrow f^2(x) = f(x^2)$ (1.2)

Now, for $f^4(x) = ((f(x))^2)^2$ by(1.2) we have;

$$= (f(x^2))^2 = (f_0 + f_1 x^2 + \cdots + f_n x^{2n})^2$$

$$= f_0^2 + (f_1 x^2)^2 + \cdots + (f_n x^{2n})^2$$

$$= f_0 + f_1 x^4 + \ldots + f_n (x^4)^n = f(x^4) \Longrightarrow f^4(x) = f(x^4),$$

etc.

So for any $i \geq 0$, $[f(x)]^{2^i} = f(x^{2^i})$. ■

1.8 Construction of Galois Field $GF(2^n)$

In this section we construct the Galois Field $GF(2^n)$ of 2^n elements $(n > 1)$ from the binary field $GF(2)$. We begin with the two elements 0 and 1, from $GF(2)$ and a new symbol α. Then we define a multiplication $(.)$ to introduce a sequence of powers of α as follows:

$\alpha^2 = \alpha.\alpha,$

$\alpha^3 = \alpha.\alpha.\alpha,$

$\vdots$

$\alpha^j = \underbrace{\alpha.\alpha \ldots \alpha}_{j-times},$

$\vdots$

Now, we have the following set of elements: $GF(2^n) = \{0, 1, \alpha, \alpha^2, \alpha^3, \ldots, \alpha^j, \ldots\}$.

Now suppose $p(x)$ is a primitive polynomial of degree n over $GF(2)$ such that $p(\alpha) = 0$. Then $p(x)$ divides $x^{2^n-1} + 1$, and so we have:

$x^{2^n-1} + 1 = q(x).p(x)$. If we replace x by α, we obtain:

$\alpha^{2^n-1} + 1 = q(\alpha)p(\alpha) = q(\alpha).0 = 0,$

This implies: $\alpha^{2^n-1} + 1 = 0,$

Adding 1 to both sides (use modulo-2 addition): $\alpha^{2^n-1} = 1$, and hence $\alpha^{2^n} = \alpha$. Therefore, the set above becomes finite and consist of the 2^n elements: $GF(2^n) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^n-2}\}$.

Remark 1.10:

(i) In the construction of the Galois field $GF(2^n)$, we use a primitive polynomial $p(x)$ of degree n and require that the element α be a root of

$p(x)$. Since the powers of α generate all the nonzero elements of $GF(2^n)$, α is a primitive element.

(ii) The elements of $GF(2^n)$ have three representations shown in Table 4.

Example 1.17: Let $n = 3$, the polynomial $p(x) = 1 + x + x^3$ is a primitive polynomial over $GF(2)$. Set $p(\alpha) = 1 + \alpha + \alpha^3 = 0$. Then $\alpha^3 = 1 + \alpha$. Using this, we can construct:

$GF(2^3) = \{0, 1, \alpha, \alpha^2, \ldots, \alpha^{2^3-2}\} = \{0, 1, \alpha, \alpha^2, \alpha^3, \alpha^4, \alpha^5, \alpha^6\}$.

The element $\alpha^3 = 1 + \alpha$ is used repeatedly to form the polynomial representations for the elements of $GF(2^3)$:

$\alpha^4 = \alpha.\alpha^3 = \alpha(1 + \alpha) = \alpha + \alpha^2$,

$\alpha^5 = \alpha.\alpha^4 = \alpha(\alpha + \alpha^2) = \alpha^2 + \alpha^3 = \alpha^2 + 1 + \alpha = 1 + \alpha + \alpha^2$,

$\alpha^6 = \alpha.\alpha^5 = \alpha(1 + \alpha + \alpha^2) = \alpha + \alpha^2 + \alpha^3 = \alpha + \alpha^2 + 1 + \alpha = 1 + \alpha^2$

Table 4: Three Representations for the Elements of $GF(2^3)$ Generated by $p(x) = 1 + x + x^3$

Power representation	Polynomial representation in α	3-*tuple* representation
0	0	(0 0 0)
1	1	(1 0 0)
α	α	(0 1 0)
α^2	α^2	(0 0 1)
α^3	$1 + \alpha$	(1 1 0)
α^4	$\alpha + \alpha^2$	(0 1 1)
α^5	$1 + \alpha + \alpha^2$	(1 1 1)
α^6	$1 + \alpha^2$	(1 0 1)

Remark 1.11:

(i) The power representation is used in multiplying or dividing the elements of $GF(2^3)$ as:

$$\alpha^i.\alpha^j = \alpha^{i+j} = \begin{cases} \alpha^{i+j} & ;i+j < 2^n-1 \\ \alpha^{i+j-(2^n-1)} & ;i+j > 2^n-1 \\ 1 & ;i+j = 2^n-1 \end{cases}$$

and, $\frac{\alpha^i}{\alpha^j} = \alpha^i.\alpha^{2^n-j-1}$ where α^{2^n-j-1} is the multiplicative inverse of α^j.

(ii) An $n-tuple$ representation is used for adding the elements of $GF(2^3)$ by adding the corresponding components of their n –tuples, in modulo-2 addition.

For example, if $u = (u_0, u_1, \ldots, u_{n-1})$ and $v = (v_0, v_1, \ldots, v_{n-1})$ in $GF(2^n)$, then $u + v = (u_0 + v_0, u_1 + v_1, \ldots, u_{n-1} + v_{n-1})$, where $u_i + v_i$ is carried out in modulo-2 addition.

Definition 1.16: Let $(E, +, .)$ be a field, and let F be a nonempty subset of E. Then F is called a subfield if $(F, +, .)$ is itself a field.

Definition 1.17: If F is a subfield of a field E, then E is called an extension field *of* F or simply an extension *of* F.

Note that, the set $GF(2^n)$ is an extension field of $GF(2)$ because $GF(2)$ is a subfield or the ground field of $GF(2^n)$.

And polynomials with coefficients from $GF(2)$ may not have roots from $GF(2)$ but has roots from an extension field of $GF(2)$.

For example, $p(x) = 1 + x + x^3$ is irreducible over $GF(2)$ and therefore it does not have roots from $GF(2)$. However, it has three roots from the field $GF(2^3)$. If we substitute the elements of $GF(2^3)$ given by (Table 4) into $1 + x + x^3$, we find that $\alpha, \alpha^2, \text{and } \alpha^4$ are the roots of $1 + x + x^3$. We may verify this as follows using (Table 4):

$$\begin{aligned} &1 + \alpha + \alpha^3 = 1 + \alpha + 1 + \alpha = 0, \\ &1 + \alpha^2 + (\alpha^2)^3 = 1 + \alpha^2 + \alpha^6 = 1 + \alpha^2 + 1 + \alpha^2 = 0, \\ &1 + \alpha^4 + (\alpha^4)^3 = 1 + \alpha + \alpha^2 + 1 + \alpha + \alpha^2 = 0. \end{aligned} \tag{1.3}$$

Since $\alpha, \alpha^2, \text{and } \alpha^4$ are all roots of $p(x)$, then

$p(x) = (x + \alpha)(x + \alpha^2)(x + \alpha^4)$. We may verify this equality by multiplying out the product above using Table 4.

Let $f(x)$ be a polynomial of degree n with coefficients from $GF(2)$. If β is a root of $f(x)$, the polynomial $f(x)$ may have other roots from $GF(2^n)$. Then what are these roots? This is answered by the following theorem.

Theorem 1.10: Let $f(x)$ be a polynomial with coefficients from $GF(2)$, and β be an element in an extension field $GF(2^n)$ of $GF(2)$ such that β is a root of $f(x)$, then for any $i \geq 0$, we have β^{2^i} is also a root of $f(x)$.

Proof: By substituting β into the equation in (1.1) we have: $[f(\beta)]^{2^i} = f(\beta^{2^i})$,

But $f(\beta) = 0 \Rightarrow f\left(\beta^{2^i}\right) = 0 \Rightarrow \beta^{2^i}$ is a root of $f(x)$ for any $i \geq 0$. ■

The element β^{2^i} is called a conjugate *of* β.

Fact 1.3: According to Theorem 1.10 above, if α is a primitive element then all conjugates of α are also primitive elements of $GF(2^n)$.

Example 1.18: The polynomial $p(x) = 1 + x + x^3$ in Example 1.17 has α as a root in the extension field $GF(2^3)$ of $GF(2)$, such that:

$p(\alpha) = 1 + \alpha + \alpha^3 = 1 + \alpha + 1 + \alpha = 0$

By using (Table 1.6) the conjugates of α are:

$(\alpha)^{2^1} = \alpha^2, \quad (\alpha)^{2^2} = \alpha^4$

[Note that $(\alpha)^{2^3} = \alpha^8 = \alpha^7.\alpha = 1.\alpha = \alpha$]. From the Theorem 1.10 these conjugates of α must be also roots of $p(x) = 1 + x + x^3$. (See eq. (1.3))

Theorem 1.11: The $2^n - 1$ nonzero elements of $GF(2^n)$ form all the roots of $x^{2^n-1} + 1$.

Proof: Let β be a non zero element in the field $GF(2^n)$. Then by using property (iv) of the fields we have: $\beta^{2^n-1} = 1$,

Adding 1 to both sides: $\beta^{2^n-1} + 1 = 0$

This implies that β is a root of $x^{2^n-1} + 1$. ■

Corollary 1.1: The elements of $GF(2^n)$ form all the roots of $x^{2^n} + x$.

Definition 1.18: A minimal polynomial *of* β *over* $GF(2)$ is a smallest degree polynomial $\varphi(x)$ such that $\varphi(\beta) = 0$, where $\beta \in GF(2^n)$.

For example, the minimal polynomial of the zero element 0 of $GF(2^n)$ is x and the minimal polynomial of the unit element 1 is $x + 1$.

Basic properties of minimal polynomials:

(i) Let $f(x)$ be a polynomial over $GF(2)$ and $\varphi(x)$ be the minimal polynomial of β. If β is a root of $f(x)$, then $f(x)$ is divisible by $\varphi(x)$.

Proof: Dividing $f(x)$ by $\varphi(x)$, we obtain $f(x) = a(x)\varphi(x) + r(x)$, where the degree of $r(x)$ is less than the degree of $\varphi(x)$.

Consider, $f(\beta) = a(\beta)\varphi(\beta) + r(\beta)$

$$0 = a(\beta).0 + r(\beta) \qquad \text{(Because } f(\beta) = \varphi(\beta) = 0\text{)}$$

$\Rightarrow r(\beta) = 0$. Now, if $r(x) \neq 0$ then r is a polynomial with degree less than degree of $\varphi(x)$ and has β as a root. This is a contradiction to the fact that $\varphi(x)$ is the minimal polynomial of β. Hence

$r(x) = 0, f(x) = a(x)\varphi(x)$ and so $f(x)$ is divisible by $\varphi(x)$. ■

(ii) The minimal polynomial $\varphi(x)$ of β in GF(2^n) is unique

Proof: Let $\gamma(x)$ & $\varphi(x)$ be two minimal polynomials of β.

If we take $\gamma(x)$ as minimal polynomials of β then by (i) $\varphi(x)$ is divisible by $\gamma(x)$. And if we take $\varphi(x)$ as minimal polynomials of β then we have $\gamma(x)$ is divisible by $\varphi(x)$. Hence, $\gamma(x) = \varphi(x)$ ■

(iii) The minimal polynomial $\varphi(x)$ of β in GF(2^n) is irreducible

Proof: Suppose that $\varphi(x)$ is not irreducible $\Rightarrow \varphi(x) = \gamma(x)\tau(x)$, where both $\gamma(x), \tau(x)$ have degrees greater than zero and less than the degree of $\varphi(x)$.

Since $\varphi(\beta) = \gamma(\beta)\tau(\beta) = 0 \Longrightarrow \gamma(\beta) = 0 \; or \; \tau(\beta) = 0$. This is a contradiction to the fact that $\varphi(x)$ is the minimal polynomial of β. Therefore, $\varphi(x)$ must be irreducible. ■

(iv) The minimal polynomial $\varphi(x)$ of β in GF(2^n) divides $x^{2^n} + x$

Proof: It follows from corollary 1 and property i. ■

(v) Let $f(x)$ be an irreducible polynomial over $GF(2)$ and $\varphi(x)$ be the minimal polynomial of β in $GF(2^n)$. If $f(\beta) = 0$, then $\varphi(x) = f(x)$.

Proof: It follows from (i) that $\varphi(x)$ divides $f(x)$.

$\Longrightarrow f(x) = a(x)\varphi(x)$,

But $\varphi(x) \neq 1$ and $f(x)$is irreducible hence $a(x) = 1 \Longrightarrow$ we must have $f(x) = \varphi(x)$ ■

(vi) Let $\varphi(x)$ be the minimal polynomial of an element β in $GF(2^n)$ and let e be the smallest integer such that $\beta^{2^e} = \beta$.

Then: $\varphi(x) = \prod_{i=0}^{e-1}(x + \beta^{2^i})$. (1.4)

Example 1.19: Consider the Galoise field $GF(2^3)$ given by table 4. Let $\varphi(x)$ be the minimal polynomial of an element $\beta = \alpha^3$.

The conjugates of β are; $(\alpha^3)^{2^1} = \alpha^6, (\alpha^3)^{2^2} = \alpha^{12} = \alpha^5$.

(Note that $(\alpha^3)^{2^3} = (\alpha^3)^8 = \alpha^{24} = \alpha^3$).

Hence, by (1.4) we have:

$\varphi(x) = \prod_{i=0}^{e-1}(x + \beta^{2^i})$ where $e = 3$

$\varphi(x) = (x + \alpha^3)(x + \alpha^5)(x + \alpha^6)$ (1.5)

Multiplying out right-hand side of the equation 1.5 using Table 4, then we obtain the following:

$$\varphi(x) = x^3 + (\alpha^5 + \alpha^3 + \alpha^6)x^2 + (\alpha^8 + \alpha^{11} + \alpha^9)x + \alpha^{14}$$

$$= x^3 + (1 + \alpha + 1 + \alpha + \alpha^2 + 1 + \alpha^2)x^2 + (\alpha + \alpha + \alpha^2 + \alpha^2)x+1$$

$$= x^3 + x^2 + 1$$

All the minimal polynomials of the elements in $GF(2^3)$ are given in the following table.

Table 5: Minimal Polynomials of the Elements in $GF(2^3)$ Generated by $p(x) = 1 + x + x^3$

Conjugate roots	Minimal polynomials
0	x
1	$1 + x$
$\alpha, \alpha^2, \alpha^4$	$1 + x + x^3$
$\alpha^3, \alpha^5, \alpha^6$	$1 + x^2 + x^3$

1.9 Vector Spaces over Finite Fields

Definition 1.19: Let V be set of elements on which a binary operation called addition $+$ is defined. Let F be a finite field. A scalar multiplication operation denoted by"." is defined between the elements in F and elements in V. Then $< V, +, >$ is called a vector space over the field F if it satisfies the following conditions:

(i) V is a commutative group under addition.

(ii) For any scalar a in F and any element v in V,$a.v$ is an element in V.

(iii) (Distributive laws) for any elements u, v in V and any scalars a, b in F,

$$a.(u+v) = a.u + a.v,$$

$$(a+b).v = a.v + b.v$$

(iv) (Associative law) for any v in V and any a and b in F,

$$(a.b).v = a.(b.v)$$

(v) Let 1 be the identity element of F. Then, for any v in V, $1.v = v$.

The elements of V are called *vectors*.

Remark 1.12: Let K be an extension field of k, then K can be considered as a vector space over k.

Since the set $GF(2^n)$ is an extension field of $GF(2)$, then $GF(2^n)$ can be considered as a vector space over $GF(2)$. Let V_n denote that set of all 2^n distinct n –tuples $(f_0 \ f_1 \ \dots \ f_{n-1})$ over $GF(2)$.

Then $(V_n, +, .)$ is a vector space with + is vector addition and . is scaller multiplication.

Example 1.20: Let $n = 3$. The vector space V_3 of all 3 –tuples over $GF(2)$ consists of the following 8 vectors: $(0\ 0\ 0), (0\ 0\ 1), (0\ 1\ 0), (0\ 1\ 1),$ $(1\ 0\ 0), (1\ 0\ 1), (1\ 1\ 0), (1\ 1\ 1)$.

Definition 1.20: Let S be a nonempty subset of a vector space V over a field F then S is a subspace of V if S is itself a vector space over F.

Theorem 1.12: Let S be a nonempty subset of a vector space V over a field F. Then S is a subspace of V if and only if the following condition is satisfied: if $u, v \in S$ and $\gamma, \delta \in F$, then $\gamma u + \delta v$ is also in S.

Note that, a necessary and sufficient condition for a nonempty subset S of a vector space V over $GF(2)$ to be a subspace is:

if $x, y \in S$, then $x + y \in S$.

Example 1.21: Consider the vector space V_3 of all 3 –tuples over $GF(2)$ given in Example 1.20. Then the set of these vectors $(0\ 0\ 0), (0\ 0\ 1), (1\ 1\ 0), (1\ 1\ 1)$ satisfies the condition of theorem 1.12, so it is a subspace of V_3.

Definition 1.21: Let $v_1, v_2, \ldots, v_k$ be k vectors in a vector space V over a field F. A vector v in V is called a linear combination $o\ f\ v_1, v_2, \ldots, v_k$ if: there are a scalars $c_1, c_2, \ldots, c_k$ in F s.t., $v = c_1 v_1 + c_2 v_2 + \ldots + c_k v_k$.

Clearly, the sum of two linear combinations of $v_1, v_2, \ldots, v_k$ is also a linear combination of $v_1, v_2, \ldots, v_k$ and the product of a scalar c in F and a linear combinations of $v_1, v_2, \ldots, v_k$ is also a linear combination of $v_1, v_2, \ldots, v_k$.

So the set of all linear combinations of $v_1, v_2, \ldots, v_k$ forms a subspace of V.

Definition 1.22: Let V be a vector space over a field F and let $S = \{v_1, v_2, \ldots, v_k\}$ be a nonempty subset of V. The span of S is defiend as: $< S >= \{c_1 v_1 + c_2 v_2 + \ldots + c_k v_k : c_i \in F, v_i \in S\}$.

Clearly, the set $< S >$ is a subspace of V, called the subspace generated (or spanned) by S. Given a subspace C of V, a subset S of C is called a generating set (or spanning set) of C if $C = < S >$ and we also say S spans C.

Remark 1.13: If S is already a subspace of V, then $< S >= S$.

Example 1.22: Let $S = \{(0\ 0\ 0\ 1), (0\ 0\ 1\ 0), (0\ 1\ 0\ 0)\}$ be a subset of V_4 over $GF(2)$. Then $< S >= \{(0\ 0\ 0\ 0), (0\ 0\ 0\ 1), (0\ 0\ 1\ 0), (0\ 1\ 0\ 0), (0\ 0\ 1\ 1), (0\ 1\ 0\ 1), (0\ 1\ 1\ 0), (0\ 1\ 1\ 1)\}$.

Definition 1.23: The vectors $v_1, v_2, \ldots, v_k$ in a vector space V over a field F are said to be *linearly dependent* if there exist constants $c_1, c_2, \ldots, c_k$ from F, not all zero, such that:

$c_1 v_1 + c_2 v_2 + \ldots + c_k v_k = 0.$ (1.6)

Otherwise, $v_1, v_2, \ldots, v_k$ are called linearly independent. That is $v_1, v_2, \ldots, v_k$ are linearly independent if whenever $c_1 v_1 + c_2 v_2 + \ldots + c_k v_k = 0$, we must have: $c_1 = c_2 = \cdots = c_k = 0$.

Example 1.23: The set $\{(0\ 0\ 0\ 1), (0\ 0\ 1\ 0), (0\ 1\ 0\ 0)\}$ is linearly independent.

Example 1.24: The set $\{(0\ 0\ 0\ 1),(1\ 0\ 0\ 0),(1\ 0\ 0\ 1)\}$ is linearly dependent since $(0\ 0\ 0\ 1)+(1\ 0\ 0\ 0)+(1\ 0\ 0\ 1)=(0\ 0\ 0\ 0)$.

Remark 1.14: Any set which contains 0 is linearly dependent. Any set containing at least two identical vectors is also linearly dependent. For example, the set $\{(0\ 0\ 0\ 1),(0\ 0\ 0\ 0),(1\ 0\ 0\ 1)\}$ and the set $\{(1\ 0\ 0\ 1),(1\ 0\ 0\ 0),(1\ 0\ 0\ 1)\}$ are linearly dependent.

Theorem 1.13: For any vector space there exists at least one linearly independent set which spans the space. Hence we have the following definition.

Definition 1.24: The set $B=\{v_1,v_2,\ldots,v_k\}$ of vectors in a vector space V over a field F is said to form a *Basis* for V if:

(i) B spans V,

(ii) B is linearly independent.

Definition 1.25: The dimension of a vector space V, denoted $\dim(V)$, is the number of vectors in a basis of V.

Remark 1.15:

(i) If $v_1,v_2,\ldots,v_k$ form a basis for a vector space V, then they must be nonzero distinct vectors.

(ii) A vector space V over a finite field F can have many bases; but all bases contain the same number of elements, called $\dim(V)$.

Theorem 1.14: If $S = \{v_1, v_2, \ldots, v_k\}$ form a basis for a vector space V, then every vector in V can be written in one and only one way as a linear combination of the vectors in S.

Example 1.25: Consider the vector space V_3 of all 3-tuples over GF(2). Let us form the following 3-tuples: $e_0 = (1\ 0\ 0)$, $e_1 = (0\ 1\ 0)$, $e_2 = (0\ 0\ 1)$. Then every 3-tuple $(a_0\ a_1\ a_2)$ in V_3 can be expressed as a linear combination of e_0, e_1, e_2 as follows:

$(a_0\ a_1\ a_2) = a_0 . e_0 + a_1 . e_1 + a_2 . e_2$

Therefore, e_0, e_1, e_2 span the vector space V_3.

We also see that e_0, e_1, e_2 are linearly independent. Hence, they form a basis for V_3 and the dimension of V_3 is 3.

This set of vectors is called the standard basis for V_3.

Theorem 1.15: Let V be a vector space over $GF(2)$ and $dim(V) = k$, then V has 2^k elements.

Proof:

(i) If $B = \{v_1, v_2, \ldots, v_k\}$ is a basis for V, then

$V = \{c_1 v_1 + c_2 v_2 + \ldots + c_k v_k : c_i \in GF(2), v_i \in B\}$.

Since $|GF(2)| = 2$, there are exactly 2 choices for each $c_i, 1 \leq i \leq k$. Hence, V has exactly 2^k elements.

Theorem 1.16: If $S = \{v_1, v_2, \ldots, v_k\}$ is linearly independent then $< S >$ is a k –dimensional subspace of V_n.

Corollary 1.2: Let V be an n –dimensional vector space, and let $B = \{v_1, v_2, \ldots, v_n\}$ be a set of n –vectors in V then:

(i) If B is linearly independent, then it is a basis for V.

(ii) If B spans V, then it is a basis for V.

Definition 1.26: Let $u = (u_0\ u_1 \ldots u_{n-1})$ and $v = (v_0\ v_1\ \ldots\ v_{n-1})$ be two n-tuples in V_n over $GF(2)$ then:

(i) We define the Euclidean inner product (also know as scalar product or dot product) of *u* and *v* as:

$u.v = u_0.v_0 + u_1.v_1 + \cdots + u_{n-1}.v_{n-1}$.

(ii) The two vectors *u* and *v* are said to be orthogonal if $u.v = 0$.

(iii) Let S be a nonempty subset of V_n. The orthogonal complement $S^{\perp}$ of S is defined to be: $S^{\perp} = \{v \in V_n : v.s = 0\ \forall\ s \in S\}$.

Example 1.26: Let $u = (1\ 1\ 1\ 1)$, $v = (1\ 1\ 1\ 0)$, $w = (1\ 0\ 0\ 1)$ be vectors in V_4 over $GF(2)$ then:

$u.v = 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 0 = 1$

$u \cdot w = 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 + 1 \cdot 1 = 0$

$v \cdot w = 1 \cdot 1 + 1 \cdot 0 + 1 \cdot 0 + 0 \cdot 1 = 1$. Hence, u and w are orthogonal.

Example 1.27: Let $S = \{(0\ 1\ 0\ 0), (0\ 1\ 0\ 1)\} \in V_4$ over $GF(2)$. To find $S^{\perp}$. Let $v = (v_0\ v_1\ v_2\ v_3) \in S^{\perp}$ then:

$v \cdot (0\ 1\ 0\ 0) = 0 \Rightarrow v_1 = 0$ and $v \cdot (0\ 1\ 0\ 1) = 0 \Rightarrow v_1 + v_3 = 0$

Hence, we have $v_1 = v_3 = 0$. Since v_1 and v_3 can be either 0 or 1, we can conclude that $S^{\perp} = \{(0\ 0\ 0\ 0), (0\ 0\ 1\ 0), (1\ 0\ 0\ 0), (1\ 0\ 1\ 0)\}$.

Theorem 1.17: Let C be a subspace of V_n. Then:

(i) $C^{\perp}$ is a subspace of V_n.

(ii) $C \cap C^{\perp} = \{0\}$

(iii) $C \cup C^{\perp} = V_n$

Theorem 1.18: Let C be a k −dimensional subspace of V_n. Then we have

$dim(C) + dim(C^{\perp}) = n.$

Remark 1.16: If $C^{\perp}$ is an orthogonal complement of C, then C is also an orthogonal complement of $C^{\perp}$. Hence, we say that C and $C^{\perp}$ are orthogonal complements.

Remark 1.17: If A is a given kxn matrix, we associate the following four fundamental vector spaces with A: the null space of A, the row space of A, the null space of A^T and the column space of A.

Remark 1.18: Recall that

(a) The Null space of $A = \{x \in R^n : AX = 0\}$

(b) The row space of A is the set of all linear combinations of the rows of $A_{k\times n}$

Theorem 1.19: If A is a given kxn matrix, then:

(i) The null space of A is the orthogonal complement of the row space of A with dim (row space) + dim (null space) $= n$.

(ii) The null space of A^T is the orthogonal complement of the column space of A with dim (column space) + dim (null space of A^T) $= k$

Most information given in this chapter are from [3,9 &11].

Chapter 2

Linear Block Codes

2.1 Basic Concepts of Block Codes

The data of output of the source encoder are represented by sequence of binary digits, zeros or ones. In block coding this sequence is segmented into message blocks $u = (u_0\, u_1\, \ldots\, u_{k-1})$ consisting of k digits each. There are a total of 2^k distinct messages. The channel encoder, according to certain rules, transforms each input message u into a word $v = (v_0\, v_1\, \ldots v_{n-1})$ with $n \geq k$.

Definition 2.1: Given the binary field $GF(2) = \{0, 1\}$, we define:

(i) A binary word w of length n over $GF(2)$ is an n–tuple $w = (w_0 w_1 \ldots w_{n-1})$ of binary digits $w_i \in GF(2)\ \forall\, i = 0, \ldots, n-1$.

(ii) A binary block code of length n over $GF(2)$ is a nonempty set C of binary words w of length n each.

(iii) Each element w of C is called a codeword in C.

(iv) The size of C, denoted by $|C|$, is the number of codewords in C.

Example 2.1: Let $C = \{0\ 0, 0\ 1, 1\ 0, 1\ 1\}$. Then C is a binary block code of length $n = 2$ and size $|C| = 4$.

A set of 2^k distinct codewords w of length n each, over the binary field $GF(2) = \{0, 1\}$, is called a *Binary Block Code* $C(n, k)$.

2.2 Definitions & Properties of the Linear Block Codes

We now introduce linear codes and discuss some of their elementary properties.

Definition 2.2: A binary block code $C(n,k)$ of length n and 2^k codewords is called linear if its 2^k codewords form a k-dimensional subspace of the vector space V_n of all n-tuples over the field $GF(2)$.

It is clear, from the above definition, a linear combination of codewords in C is also a codeword in C.

Basic properties of a linear block code $C(n,k)$:

(i) The zero word $(0\ 0\ \ldots\ 0)$, is always a codeword.

(ii) If c is a codeword, then $(-c)$ is also a codeword.

(iii) A linear code is invariant under translation by a codeword. That is, if c is a codeword in linear code C, then $C + c = C$.

(iv) The dimension k of the linear code $C(n,k)$ is the dimension of C as a subspace of V_n over $GF(2)$, i.e., $dim(C) = k$.

Example 2.2: Let $C = \{(\lambda\ \lambda \ldots\ \lambda) : \lambda \in GF(2)\}$. Then C is a linear block code often called the repetition code.

Example 2.3: Let $C = \{(1\ 1\ 1\ 0\ 0), (0\ 0\ 1\ 1\ 0), (1\ 1\ 1\ 1\ 1), (1\ 1\ 0\ 1\ 0), (0\ 0\ 0\ 1\ 1), (1\ 1\ 0\ 0\ 1), (0\ 0\ 0\ 0\ 0), (0\ 0\ 1\ 0\ 1)\}$. Then C is a $(5,3)$ linear block code. Since any linear combination of the codewords in C is also a codeword in C. For instance,

$$(0\ 0\ 0\ 1\ 1)+(1\ 1\ 0\ 0\ 1)=(1\ 1\ 0\ 1\ 0)\in C$$

$$(0\ 0\ 0\ 1\ 1)+(1\ 1\ 0\ 1\ 0)=(1\ 1\ 0\ 0\ 1)\in C$$

$$(1\ 1\ 0\ 0\ 1)+(1\ 1\ 0\ 1\ 0)=(0\ 0\ 0\ 1\ 1)\in C$$

$$(0\ 0\ 0\ 1\ 1)+(1\ 1\ 0\ 0\ 1)+(1\ 1\ 0\ 1\ 0)=(0\ 0\ 0\ 0\ 0)\in C$$

Example 2.4: Find a basis for the linear block code $C(5,3)$ given in Example 2.3.

To find a basis for C, we use algorithm 1 of Appendix B as follows;

$$A=\begin{bmatrix}1&0&1&1&0&1&0\\1&0&1&1&0&1&0\\1&1&1&0&0&0&1\\0&1&1&1&1&0&0\\0&0&1&0&1&1&1\end{bmatrix}\xrightarrow{REF}\begin{bmatrix}1&0&1&1&0&1&0\\0&1&0&1&0&1&1\\0&0&1&0&1&1&1\\0&0&0&0&0&0&0\\0&0&0&0&0&0&0\end{bmatrix}$$

The leading 1's in the REF are in 1, 2 & 3, then the original columns of A corresponding to these leading columns form a basis B for C. Thus $B=\left\{\begin{bmatrix}1\\1\\1\\0\\0\end{bmatrix},\begin{bmatrix}0\\0\\1\\1\\0\end{bmatrix},\begin{bmatrix}1\\1\\1\\1\\1\end{bmatrix}\right\}$ is a basis of given code C.

2.3 The Generator Matrix $G_{k\times n}$

Since a linear code $C(n,k)$ is k-dimensional subspace of V_n, then knowing a basis of it enables us to describe its codewords explicitly. In coding theory, a basis for a linear code C is often represented by a matrix G, called a generator matrix. To obtain the generator matrix G for the linear $C(n,k)$ code we choose any k linearly independent codewords $g_0, g_1, \dots, g_{k-1}$ in C and arrange them as rows of a $k \times n$ matrix. So, we have

$$G = \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ . \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} \cdots & g_{0,n-} \\ g_{10} & g_{11} \cdots & g_{1,n-} \\ . & . & . \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,n} \end{bmatrix} \quad (2.1)$$

where $g_i = (g_{i0}\ g_{i1}\ \cdots\ g_{i,n-1})$ for $0 \leq i < k$.

Then each codeword v of C is a linear combination of the codewords g_i.

i.e. $v = \sum_{i=0}^{k-1} a_i g_i$ where $a_i \in \{0, 1\}$.

Definition 2.3: A generator matrix $G_{k \times n}$ for a linear $C(n, k)$ code is a matrix whose k rows form a basis for C.

By this definition, then C is the row space of G.

Algorithm 1 of Appendix B can be used to find the generator matrix G of any linear $C(n, k)$ linear code .

Example 2.5: The generator matrix for the linear code $C(5, 3)$ given in Example 2.3 is

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

Note $B = \{(1\ 1\ 1\ 0\ 0), (0\ 0\ 1\ 1\ 0), (1\ 1\ 1\ 1\ 1)\}$ is a basis of $C(5,3)$ as shown in example 2.4.

2.4 Encoding Scheme

If $u = (u_0\ u_1\ \ldots\ u_{k-1})$ is the message to be encoded, then the corresponding codeword v can be given as follows:

$$v = u.G$$

$$= (u_0\ u_1\ \dots\ u_{k-1}) \cdot \begin{bmatrix} g_0 \\ g_1 \\ . \\ . \\ . \\ g_{k-1} \end{bmatrix} = u_0 g_0 + u_1 g_1 + \cdots + u_{k-1} g_{k-1}. \qquad (2.2)$$

i.e. $\sum_{i=0}^{k-1} u_i g_i$ is a codeword of C with coefficients $u_0, u_1, \dots, u_{k-1}$.

Remark 2.1:

For each k-tuple (message) $u = (u_0, u_1, \dots, u_{k-1})$ there corresponds one and only one codeword $v = (v_0, v_1, \dots v_{n-1})$. So there are 2^k distinct messages and corresponding 2^k distinct codewords.

Example 2.6: Let $U = \{(0\ 0\ 0), (0\ 1\ 1), (1\ 1\ 0), (0\ 1\ 0), (0\ 0\ 1), (1\ 0\ 0), (1\ 0\ 1), (1\ 1\ 1)\}$ be the set of messages to be encoded using the generator matrix $G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$ of the linear code $C(5,3)$ given in

Example 2.3:

Then the corresponding codewords are: $V_0, V_1, \dots V_{n-1}$

$$U_0 = (0\ 0\ 0) \xrightarrow{U_0 \cdot G} V_0 = (0\ 0\ 0\ 0\ 0)$$

$$U_1 = (0\ 1\ 1) \xrightarrow{U_1 \cdot G} V_1 = (1\ 1\ 0\ 0\ 1)$$

$$U_2 = (1\ 1\ 0) \xrightarrow{U_2 \cdot G} V_2 = (1\ 1\ 0\ 1\ 0)$$

$$U_3 = (0\ 1\ 0) \xrightarrow{U_3 \cdot G} V_3 = (0\ 0\ 1\ 1\ 0)$$

$$U_4 = (0\ 0\ 1) \xrightarrow{U_4 \cdot G} V_4 = (1\ 1\ 1\ 1\ 1)$$

$$U_5 = (1\ 0\ 0) \xrightarrow{U_5 \cdot G} V_5 = (1\ 1\ 1\ 0\ 0)$$

$$U_6 = (1\ 0\ 1) \xrightarrow{U_6 \cdot G} V_6 = (0\ 0\ 0\ 1\ 1)$$

$$U_7 = (1\ 1\ 1) \xrightarrow{U_7 \cdot G} V_7 = (0\ 0\ 1\ 0\ 1)$$

Remark 2.2: Since the rows of G generate the $C(n,k)$ linear code, the encoder has only to store the k rows of G and to form a linear combination of these rows with entries from the message.

Definition 2.4: Let $u = (u_0\ u_1 \dots u_{k-1})$ be a message to be encoded. Then the corresponding codeword $v = (v_0\ v_1 \dots\ v_{n-1})$ in a linear code $C(n,k)$ has a systematic structure, if it may be divided into two parts, the message part consisting of the k digits $u_0\ u_1 \dots u_{k-1}$ and the redundant checking part which consists of $n-k$ parity-check digits as shown in Figure 2 below

Redundant Checking part n-k digits	Message Part k digits

Figure 2: Systematic Form of a Codeword

Definition 2.5: A linear systematic block code is a linear code with the systematic structure of the codewords.[15]

The encoder is called *systematic*.

Using elementary row operations and/or column permutations for a linear systematic (n,k) code the generator matrix G can be written in the following form:

$$G_{k\times n} = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{k-1} \end{bmatrix} \rightarrow \acute{G} = (P_{k\times(n-k)} | I_k) \tag{2.3}$$

$$= \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & & & \vdots & \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

where I_k is the $k \times k$ identity matrix and P is a $k \times (n-k)$ matrix which generates *parity-check digits*.

We call this form of G the *systematic form* of a generator matrix G.

Now, let $u = (u_0\ u_1\ \dots\ u_{k-1})$ be the message to be encoded using the systematic form of a generator matrix G, then the corresponding codeword is:

$$v = u.\acute{G} = u.[P\ \ I_k] = [uP\ \ u].$$

And hence, the rightmost k digits of v are represent, the message digits $u_0, u_1, \dots, u_{k-1}$ to be encoded:

$$v_{n-k+i} = u_i \quad \forall\ 0 \leq i < k \tag{2.4}$$

And the leftmost $n-k$ digits of v represent the parity-check digits, which is linear sums of the message digits

$$v_j = u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k-1} p_{k-1,j} \quad \forall\ 0 \leq j < n-k. \tag{2.5}$$

The $n-k$ equations given by (2.5) are called *parity-check equations* of the code.

Example 2.7: Consider the $C(5,3)$ given in example 2.3 with the generator matrix

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

Using elementary row operations and/or column then the generator matrix G can be written as follows:

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{(P_{2\times 2}|I_2)} \acute{G} = \left(\begin{array}{cc|ccc} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right).$$

Therefore, a linear systematic code $\grave{C}(5,3)$ which is generated by $\grave{G}$ is completely specified by (2.4) and (2.5). To show this:

Let $u = (u_0\ u_1\ u_2)$ be a message to be encoded. The corresponding codeword is $v = (v_0\ v_1\ v_2\ v_3\ v_4) = (u_0\ u_1\ u_2).\left(\begin{array}{ccc|cc} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right)$

given by these equations:

$$v_0 = u_0$$
$$v_1 = u_0 + u_1 + u_2$$
$$v_2 = u_0$$
$$v_3 = u_1$$
$$v_4 = u_2$$

Thus, the corresponding codewords for the messages $\{(0\ 0\ 0), (0\ 1\ 1), (1\ 1\ 0), (0\ 1\ 0), (0\ 0\ 1), (1\ 0\ 0), (1\ 0\ 1), (1\ 1\ 1)\}$ given in Example 2.6 are shown in Table 6 below:

Note that the code C generated by G is not necessarily the same code $\grave{C}$ that would be obtained by $\grave{G}$. But $\grave{C}$ is an *equivalent code* of C which is defined in the following definition:

Table 6: Linear Systematic block code with $k = 3$ and $n = 5$

Message u	Codeword v
(0 0 0)	(0 0 $\underbrace{0\ 0\ 0}_{u}$)
(0 1 1)	(0 0 $\underbrace{0\ 1\ 1}_{u}$)
(1 1 0)	(1 0 $\underbrace{1\ 1\ 0}_{u}$)
(0 1 0)	(0 1 $\underbrace{0\ 1\ 0}_{u}$)
(0 0 1)	(0 1 $\underbrace{0\ 0\ 1}_{u}$)
(1 0 0)	(1 1 $\underbrace{1\ 0\ 0}_{u}$)
(1 0 1)	(1 0 $\underbrace{1\ 0\ 1}_{u}$)
(1 1 1)	(1 1 $\underbrace{1\ 1\ 1}_{u}$)

Definition 2.6: Two codes C_1 and C_2 are equivalent if they can be formed by generator matrices G_1 and G_2, respectively, that are related by elementary row operations or column permutation. We call these matrices G_1 & G_2 equivalent generator matrices.

2.5 The Parity-Check Matrix $H_{(n-k)\times n}$

Another matrix associated with every linear block code is the parity-check matrix H.

By Theorem 1.19, the null space of G is orthogonal to the row space of G. So we construct an $(n-k) \times n$ matrix H whose rows form a basis of the null space of G in this case $G.H^T = 0$.

An n –tuple v is a codeword in the code generated by G if and only if $H.v^T = 0$. This matrix H is called a parity-check matrix of the code C.

The 2^{n-k} linear combinations of the rows of the matrix H form the dual code $C^{\perp}(n, n-k)$ of C which is defined as follows;

Definition 2.7: Let $C(n,k)$ be a linear code in V_n. The *dual code* $C^{\perp}$ of C is the orthogonal complement of the subspace C of V_n.

Note that $C^{\perp}$ is linear code with $\dim(C) + \dim(C^{\perp}) = n$.

Remark 2.3: The dual code $C^{\perp}$ of C is spanned by the null space of the generator matrix G of C.

Example 2.8: Consider the linear block code $C(5,3)$ given in example 2.3 with the generator matrix $G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$.

To find the parity-check matrix H of C, we find a basis of $C^{\perp}$ which forms the rows of H. To do this we use Algorithm 2 of appendix B as follows;

Form the matrix G

$$G = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \xrightarrow{RREF} \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \xrightarrow{(X_{2\times 2}|I_2)}$$

$$\begin{matrix} 1 & 2 & 3 & 4 & 5 \end{matrix}$$

$$\acute{G} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} . P^{*} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$= \left(\begin{array}{cc|ccc} 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{array}\right) \xrightarrow{(I_2|X^T)}$$

$$\begin{matrix} 2 & 5 & 1 & 3 & 4 \end{matrix}$$

$$\acute{H} = \left(\begin{array}{cc|ccc} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 \end{array}\right) \rightarrow H = \acute{H}.P^{*T} = \left(\begin{array}{cc|ccc} 1 & 0 & \mathbf{1} & 0 & 0 \\ 0 & 1 & \mathbf{1} & 1 & 1 \end{array}\right).\begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$

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

Therefore, $C^{\perp} = \{(1\,1\,0\,0\,0), (1\,0\,1\,1\,1), (0\,0\,0\,0\,0), (0\,1\,1\,1\,1)\}$.

As a result, a parity-check matrix $H_{(n-k)\times n}$ for a linear code $C(n,k)$ is a generator matrix for its dual code $C^{\perp}$ where

$$H = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,n-1} \\ h_{10} & h_{11} & \cdots & h_{1,n-1} \\ \vdots & \vdots & \cdots & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix}$$

For a linear systematic (n,k) code C the parity check matrix can be written in systematic form as the follows:

$$H \rightarrow \acute{H} = [I_{n-k} \quad P^T] = \begin{bmatrix} \mathbf{1} & 0 & \cdots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & \cdots & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ & & & & & \vdots & & \\ 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \quad (2.6)$$

Where P^T is the transpose of the matrix P in $\acute{G} = [P_{k\times n-k} \quad I]$.

Theorem 2.1: For an (n,k) linear systematic block code C with generator matrix $\acute{G}$ and parity check matrix $\acute{H}$ we have $\acute{G}.\acute{H}^T = 0$.

Proof: Consider $\acute{G} = \left(P_{k\times(n-k)} \middle| I_k\right)$ and

$\acute{H} = \left(I_{n-k} \middle| P^T{}_{(n-k)\times k}\right) \rightarrow \acute{H}^T = \begin{pmatrix} I_{n-k} \\ P_{k\times(n-k)} \end{pmatrix}$.

Now, we have

$$\acute{G}.\acute{H}^T = \left(P_{k\times(n-k)} \middle| I_k\right).\begin{pmatrix} I_{n-k} \\ P_{k\times(n-k)} \end{pmatrix} = P_{k\times(n-k)} + P_{k\times(n-k)} = 0$$

where + is modulo-2 addition. ■

Note that the matrix $\acute{H}$ given in (2.6) is a parity-check matrix of an equivalent code $\acute{C}$, generated by $\acute{G}$ given in (2.3), of the linear code C.

Theorem 2.2: An (n, k) linear systematic block code $\acute{C}$ is completely specified by its parity-check matrix $\acute{H}$.

Proof: Let $u = (u_0\, u_1 \ldots u_{k-1})$ be the message to be encoded. Then the corresponding codeword would be

$v = (v_0\, v_1 \ldots v_{n-k-1}\, u_0\, u_1 \ldots u_{k-1})$

Since $\acute{H}.v^T = 0$ then we have;

$$\begin{bmatrix} 1 & 0 & \ldots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & \ldots & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ & & & & & \vdots & & \\ 0 & 0 & \ldots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \cdot \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{n-k-1} \\ u_0 \\ u_1 \\ \vdots \\ u_{k-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix},$$

$$v_0 + p_{00}.u_0 + p_{10}.u_1 + \cdots + p_{k-1,0}.u_{k-1} = 0$$

$$v_1 + p_{01}.u_0 + p_{11}.u_1 + \cdots + p_{k-1,1}.u_{k-1} = 0 \qquad (2.7)$$

$$\vdots$$

$$v_{n-k-1} + p_{0,n-k-1}.u_0 + p_{1,n-k-1}.u_1 + \cdots + p_{k-1,n-k-1}.u_{k-1} = 0$$

These $n - k$ parity-check equations can be give by this general equation:

$$v_j + p_{0j}.u_0 + p_{1j}.u_1 + \cdots + p_{k-1,j}.u_{k-1} = 0 \qquad (2.8)$$

for $0 \leq j < n - k$.

Rearranging equation (2.8) we obtain the same parity-check equations of (2.5).

2.6 Encoding Circuit for a Linear Systematic (n, k) Code

In this section, we will introduce an $(n\,k)$ linear systematic codes via a direct descriptive approach by the implementation in (Figure 3).

Given k-data bits as the message $u = (u_0 \ \ldots \ u_{k-1})$, then the encoding circuit for an (n, k) linear systematic code can be implemented based on the equations of (2.4) and (2.5).

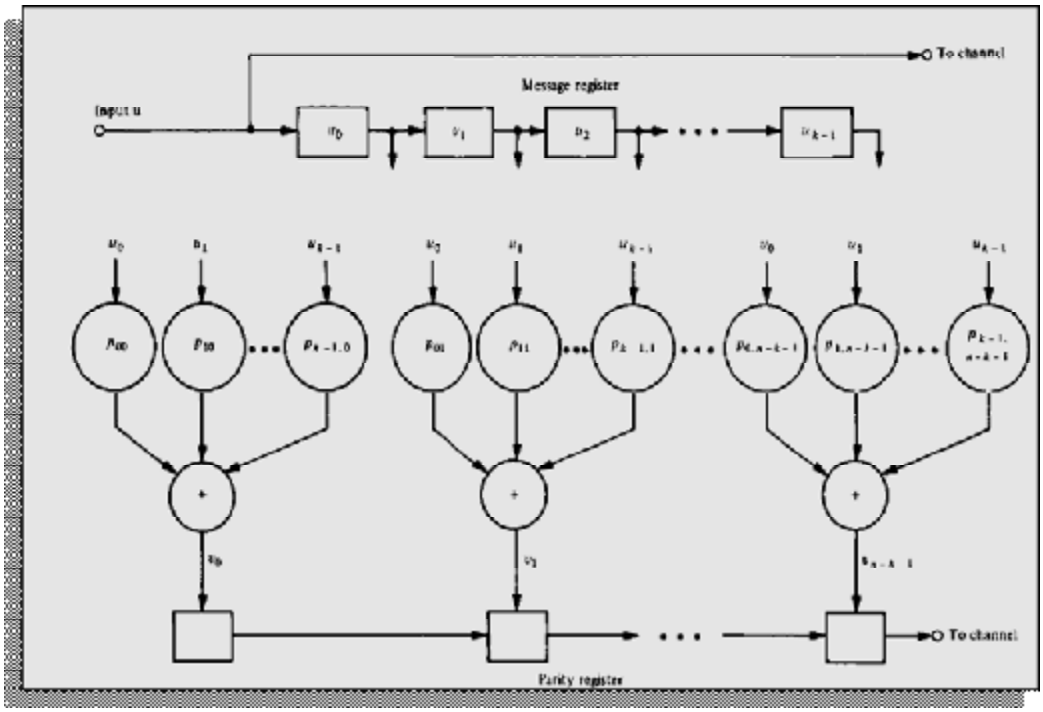


Figure 3: Encoding Circuit for a Linear Systematic (n, k) Code

Here $\oplus$ denotes modulo-2 addition and $\rightarrow$ (p_{ij}) $\rightarrow$ denotes connection if $p_{ij} = 1$ and no connection if $p_{ij} = 0$.

Let $u = (u_0 \ \ldots \ u_{k-1})$ to be encoded then this message is shifted into the message register and simultaneously into the channel. As soon as the entire message has entered the message register, the $n - k$ parity-check

digits are formed at the outputs of the $n-k$ modulo-2 adders $\oplus$. These parity-check digits are then serialized and shifted into the channel.

Example 2.9: The encoding circuit for a linear systematic $(5,3)$ code given in (Example 2.7) is shown in (Figure 4), where the connection is based on the parity-check equations given in this example.

$$v_0 = u_0$$

$$v_1 = u_0 + u_1 + u_2$$

$$v_2 = u_0, v_3 = u_1, v_4 = u_2$$

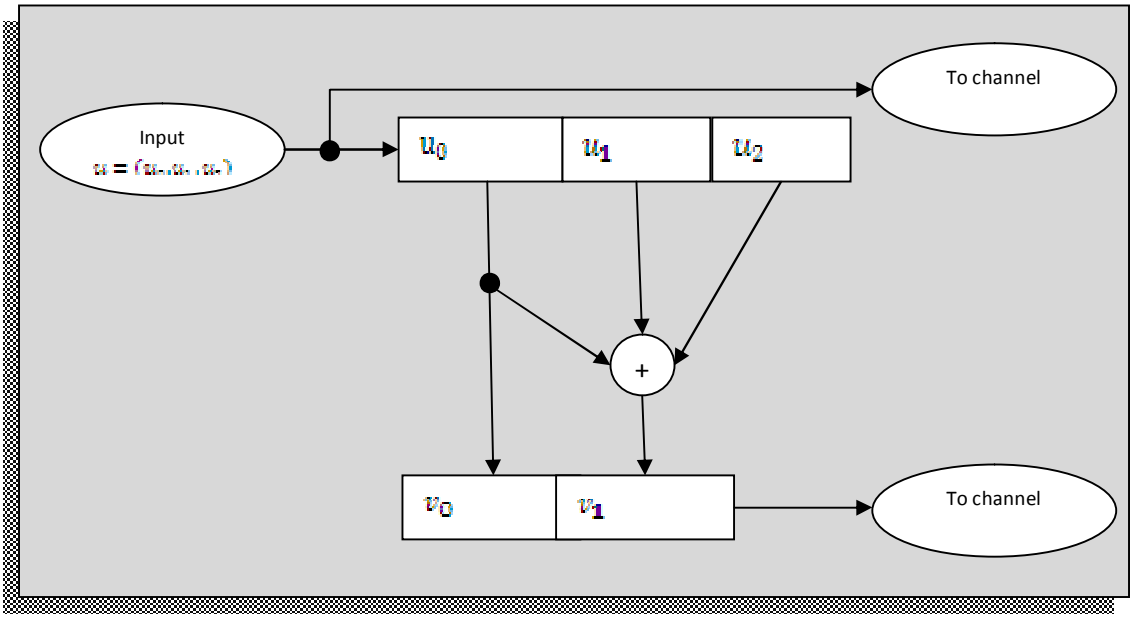


Figure 4: Encoding Circuit For a Linear Systematic $(5,3)$ Code

Chapter 3

Error Detection, Error Correction & Decoding Schemes

A fundamental concept in secure communication of data is the ability to detect and correct the errors caused by the channel. In this chapter, we will introduce the general schemes/methods of linear codes decoding.

3.1 Channel Model / Binary Symmetric Channel

The channel is the medium over which the information is conveyed. Examples of channels are telephone lines, internet cables and phone channels, etc. These are channels in which information is conveyed between two distinct places or between two distinct times, for example, by writing information onto a computer disk, then retrieving it at later time.

Definition 3.1: A communication channel consists of a finite channel alphabet $A = \{a_0, a_1, \dots, a_{q-1}\}$ as well as a set of forward channel probabilities $P(a_j \text{ received} \mid a_i \text{ sent})$, satisfying

$$\sum_{j=0}^{q-1} P(a_j \text{ received} \mid a_i \text{ sent}) = 1, \quad \forall\, 0 \leq i < q \tag{3.1}$$

See figure 5.

Note that $P(a_j \text{ received} \mid a_i \text{ sent})$ is the conditional probability that a_j is received, given that a_i is sent.

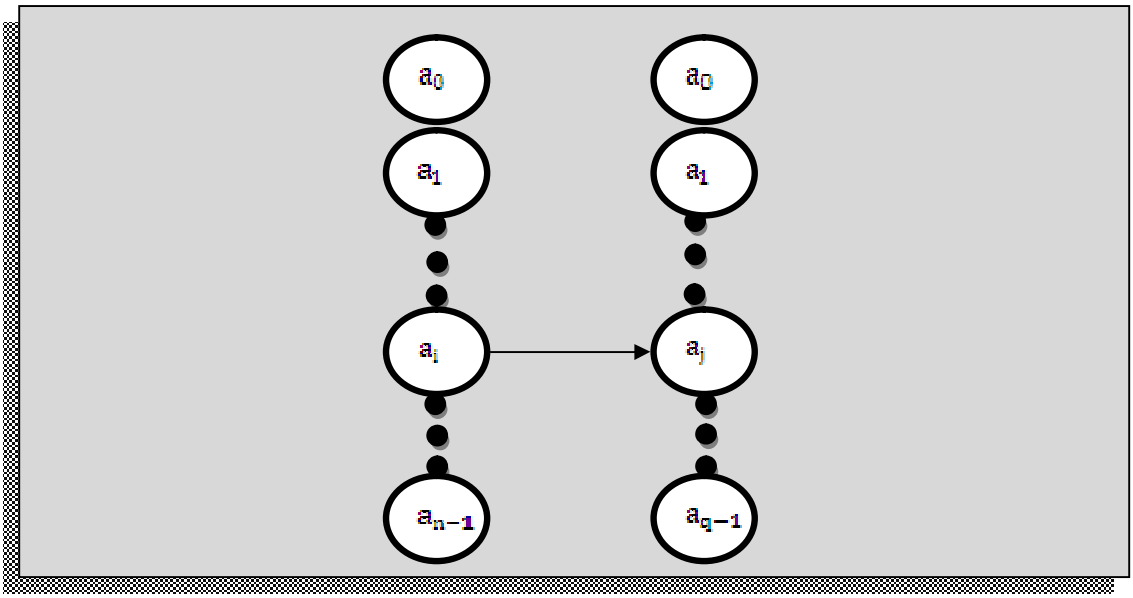


Figure 5: A Communication Channel

Definition 3.2: A communication channel is said to be memoryless if the outcome of any transmission is independent of the outcome of any previous transmission i.e.

If $v = (v_0\ v_1\ \dots\ v_{n-1})$ and $r = (r_0\ r_1\ \dots\ r_{n-1})$ are words of length n, then

$$p(r \text{ received}|v \text{ sent}) = \prod_{i=0}^{n-1} p(r_i \text{ received}|v_i \text{ sent}) \qquad (3.2)$$

Now, for purposes of analysis, channels are frequently characterized by mathematical models, which (it is hoped) are sufficiently accurate to be representative of the attributes of the actual channel.

In this thesis we restrict our work on a particularly simple and practically important channel model, called the binary symmetric channel (BSC), defined as follows:

Definition 3.3: A binary symmetric channel (BSC) is a memoryless channel which has channel alphabet $\{0, 1\}$ and channel probabilities

$$p(1 \text{ received}|0 \text{ sent}) = p(0 \text{ received}|1 \text{ sent}) = p < \frac{1}{2},$$

$$p(0 \text{ received}|0 \text{ sent}) = p(1 \text{ received}|1 \text{ sent}) = 1 - p.$$

Figure 6 below shows a BSC with crossover probability p.

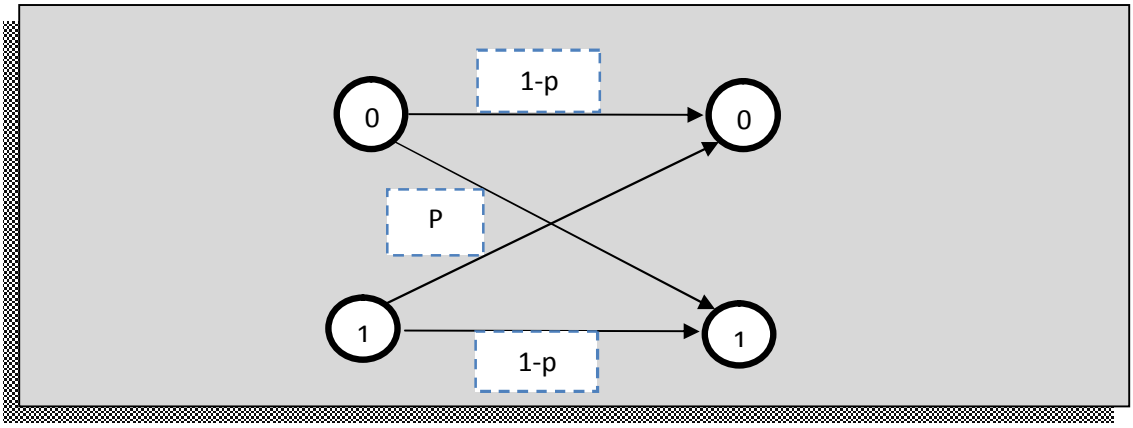


Figure 6: Binary Symmetric Channel

3.2 General Methods of Decoding Linear Codes over BSC

In a communication channel we assume a codeword $v = (v_0 \ldots v_{n-1})$ is transmitted and suppose $r = (r_0 \ldots r_{n-1})$ is received at the output of the channel. If r is a valid codeword, we may conclude that there is no error in v. Otherwise, we know that some errors have occurred and we need to find the correct codeword that was sent by using any of the following general methods of linear codes decoding:

1. Maximum likelihood decoding,
2. Nearest neighbor/Minimum distance decoding
3. Syndrome decoding
4. Standard array
5. Syndrome decoding using truth table

These methods for finding the most likely codeword sent are known as decoding methods.

Recall that the model of a data transmission system can be represented as follows in Figure 7.

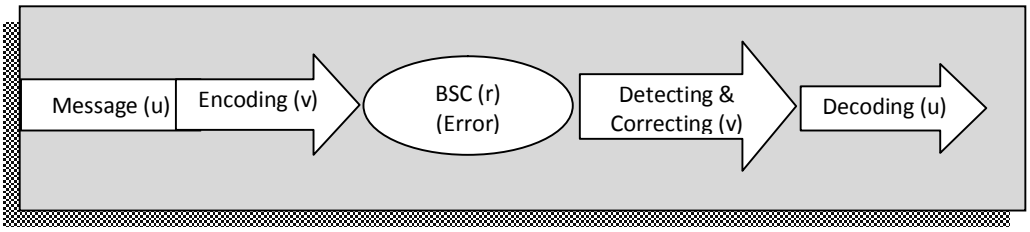


Figure 7: Simplified Model of a Code System

We start with the maximum likelihood decoding, which coincides with the minimum distance decoding under some conditions which will be studied in the following two sections, then we consider more comprehensive methods.

3.3 Maximum Likelihood Decoding

Suppose the codewords $\{v_0, v_1, \dots, v_{2^k-1}\}$ form the linear block code $C(n,k)$ and suppose a BSC with crossover probability $p < \frac{1}{2}$ is used.

Let a word $r = (r_0\ r_1\ \dots\ r_{n-1})$ of length n be received when a codeword $v_r = (v_{r_0}\ v_{r_1}\ \dots\ v_{r_{n-1}}) \in C$ is sent. Then, The maximum likelihood decoding (MLD) will conclude that v_r is the most likely codeword transmitted if v_r maximizes the forward channel probabilities i.e.

$$P(r \text{ received} | v_r \text{ sent}) = max_{v_j \in C} P(r \text{ received} | v_j \text{ sent}) \qquad (3.3)$$

$$\forall\ j = 0, 1, \dots, 2^k - 1$$

where $p(r \text{ received} | v_j \text{ sent}) = \prod_{i=0}^{n-1} p(r_i \text{ received} | v_{j_i} \text{ sent})$.

Example 3.1: Let $C = \{(0\ 0\ 0), (0\ 1\ 1)\}$ be a linear block code. Let $r = (1\ 1\ 1)$ is received when $v = (0\ 1\ 1)$ is transmitted over a BSC with crossover probability $p = 0.05$ then we can try to find the more likely codeword sent for r by computing the forward channel probabilities:

$$P(1\ 1\ 1 \text{ received}|0\ 0\ 0 \text{ sent}) =$$
$$= (P(1 \text{ received}|0 \text{ sent}))^3$$
$$= (0.05)^3 = 0.000125,$$
$$P(1\ 1\ 1 \text{ received}|0\ 1\ 1 \text{ sent}) =$$
$$= P(1 \text{ received}|0 \text{ sent}) \times (P(1 \text{ received}|1 \text{ sent}))^2$$
$$= (0.05)(0.95)^2 = 0.045125.$$

According to MLD; since the second probability is larger than the first, we can conclude that 011 is more likely to be the codeword sent.

Now, There are two kinds of MLD:

(i) Complete maximum likelihood decoding (CMLD). If a word r is received, find the most likely codeword transmitted. If there are more than one such codewords, select one of them arbitrarily.

(ii) Incomplete maximum likelihood decoding (IMLD). If a word r is received, find the most likely codeword transmitted. If there are more than one such codewords, request a retransmission.

In general; for a BSC with crossover probability $p < \frac{1}{2}$ we have the following forward channel probability:

$$P(r \text{ received}|v \text{ sent}) = \prod_{i=0}^{n-1} p(r_i \text{ received}|v_i \text{ sent})$$
$$= p^e(1-p)^{n-e} \qquad (3.4)$$

where e is the number of places at which r and v differ.

Since $p < \frac{1}{2} \Rightarrow 1 - p > p$, so this probability is larger for larger values of $n - e$, i.e. for smaller values of e.

Hence, this probability is maximized by choosing a codeword v for which e is as small as possible.

This value e leads us to another decoding method that is the nearest neighbor decoding or (minimum distance decoding).

3.4 Nearest Neighbor Decoding/Minimum Distance Decoding

In this section an important parameters of linear block codes called the hamming distance and hamming weight are introduced as well as the minimum distance decoding.

Definition 3.4: Let $x = (x_0\ x_1\ \ldots\ x_{n-1})$ and $y = (y_0\ y_1\ \ldots\ y_{n-1})$ be two binary words. The Hamming distance or simply (distance) from x to y, denoted by $d(x, y)$ or $d_H(x, y)$, is defined to be the number of positions that the corresponding elements differ:

$$d_H(x, y) = d(x, y) = \sum_{i=0}^{n-1} d(x_i, y_i), \qquad (3.5)$$

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

Example 3.2: Let $x = (0\ 0\ 1\ 1\ 1)$ and $y = (1\ 1\ 0\ 0\ 1)$ be two codewords in the linear block $C(5,2)$ over $GF(2)$. Then the hamming distance from x to y is;

$d(x, y) = \sum_{i=0}^{4} d(x_i, y_i) = 1 + 1 + 1 + 1 + 0 = 4.$

Theorem 3.1: Let x, y and z be words of length n over $GF(2)$. Then we have:

(i) $0 \leq d(x,y) \leq n$,

(ii) $d(x,y) = 0 \iff x = y$,

(iii) $d(x,y) = d(y,x)$,

(iv) $d(x,z) \leq d(x,y) + d(y,z)$ (Triangle inequality).

Proof: The proof of **(i)**, **(ii)** and **(iii)** is obvious from the definition of the Hamming distance. We just prove **(iv)**.

Use the definition of the hamming distance in (3.5). Therefore, we want to show the following:

$$d(x_0,z_0) + \cdots + d(x_{n-1},z_{n-1}) \leq$$

$$d(x_0,y_0) + \cdots + d(x_{n-1},y_{n-1}) + d(y_0,z_0) + \cdots + d(y_{n-1},z_{n-1}) \qquad (3.6)$$

If $x_i = z_i$ then $d(x_i,z_i) = 0$

$\Rightarrow d(x_i,z_i) = 0 \leq d(x_i,y_i) + d(y_i,z_i) = 0 + \mathbf{0}$ or $1 + \mathbf{1}$.

If $x_i \neq z_i$ then $d(x_i,z_i) = 1$.

If $y_i \neq x_i \Rightarrow d(y_i,x_i) = 1$. If $y_i \neq z_i \Rightarrow d(y_i,z_i) = 1$.

Otherwise, if $y_i = x_i$ and $y_i = z_i \Rightarrow x_i = z_i$ which is contradiction.

Hence, $d(x_i,z_i) \leq d(x_i,y_i) + d(y_i,z_i)$, for some $i = 0,1,\ldots,n-1$.

Therefore, (3.6) is proved $\Rightarrow (iv)$ is also proved. ■

Definition 3.5: Let $x = (x_0\, x_1 \ldots x_{n-1})$ be a binary n-tuple. The (Hamming) weight of x, denoted by $w(x)$, is defined to be the number of nonzero components of x; that is,

$$w(x) = d(x,0) = \sum_{i=0}^{n-1} d(x_i,0), \qquad (3.7)$$

where 0 is the zero word and $d(x_i, 0) = \begin{cases} 1 & if\ x_i \neq 0 \\ 0 & if\ x_i = 0 \end{cases}$

Remark 3.1: The hamming weight of x can also be equivalently defined by; $w(x) = w(x_0) + w(x_1) + \cdots + w(x_{n-1})$.

Example 3.3: The hamming weight of $x = (1\ \ 1\ \ 0\ \ 0\ \ 1)$ is 3.

Lemma 3.1: If $x, y \in V_n$, then $d(x, y) = w(x - y)$.

Proof: $d(x, y) = d(x - y, y - y)$ (since V_n is a vector space which is commutative group under addition. So for $y \in V_n$ there is additive inverse denoted by $-y \in V_n$ s.t., $-y + y = y - y = 0$)

$$= d(x - y, 0) = w(x - y)$$

$\Rightarrow d(x, y) = w(x - y)$. ■

Note that in binary codes negation is unnecessary. The following corollary is an immediate consequence of lemma 3.1.

Corollary 3.1: If x, y be two binary n –tuples, then $d(x, y) = w(x + y)$.

Example 3.4: For $x = (1\ \ 0\ \ 0\ \ 1\ \ 0\ \ 1\ \ 1), y = (1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1\ \ 0)$

$d(x, y) = 4$ and $w(x + y) = w(0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 1) = 4$.

We now explain the minimum distance decoding; suppose the codewords $v_0, v_1, \ldots, v_{2^k-1}$ from a code $C(n, k)$ are being sent over a BSC. If a word r is received, the nearest neighbor decoding or (minimum distance decoding) will decode r to the codeword v_r that is the closest one to the received word r. Such procedures can be realized by an exhaustive

search on the set of codewords which consists of comparing the received word with all codewords and choosing of the closest codeword. That is;

$$d(r, v_r) = min_{v_i \in C}\, d(r, v_i) \;\; \forall\, i = 0,1, \ldots, 2^k - 1. \qquad (3.8)$$

Just as for the case of maximum likelihood decoding, we can distinguish between complete and incomplete decoding for the nearest neighbor decoding. For a given received word r, if two or more codewords satisfy (3.8), then the complete decoding arbitrarily selects one of them to be the most likely word sent, while the incomplete decoding requests for a retransmission.

Theorem 3.2: For a BSC with crossover probability $p < \frac{1}{2}$, the maximum likelihood decoding is the same as the nearest neighbour decoding.

Proof: Let $C(n, k)$ denote the code in use and let r denote the received word (of length n). Then for any codeword v, and for any $0 \leq e \leq n$, using MLD we have

$$d(r, v) = e \Leftrightarrow P(r\ received | v\ sent) = p^e(1-p)^{n-e} \qquad (3.9)$$

Since $p < \frac{1}{2} \Longrightarrow 1 - p > p$, so the probability in (3.9) is larger for larger values of $n - e$, i.e. for smaller values of $e = d(r, v)$. Hence, it is the same as the nearest neighbor decoding. ■

Remark 3.2: In this thesis we will assume that all communication channels are binary channels having crossover probabilities $p < \frac{1}{2}$. Consequently, we can use the minimum distance decoding to perform MLD.

Example 3.5: Recall that in example 3.1 we use MLD, let us use the minimum distance decoding for the same example.

$d((1\ 1\ 1),(0\ 0\ 0)) = 3$, $d((1\ 1\ 1),(0\ 1\ 1)) = 1$, by using nearest neighbor decoding, we decode r to $(0\ 1\ 1)$. The IMLD table for C is as shown in Table 7, where '_' means that retransmission is sought.

Table 7: IMLD Table for C.

Received r	$d(r,(0\ 0\ 0))$	$d(r,(0\ 1\ 1))$	Decode to
$(0\ 0\ 0)$	0	2	$(0\ 0\ 0)$
$(1\ 0\ 0)$	1	3	$(0\ 0\ 0)$
$(0\ 1\ 0)$	1	1	_
$(0\ 0\ 1)$	1	1	_
$(1\ 1\ 0)$	2	2	_
$(0\ 1\ 1)$	2	0	$(0\ 1\ 1)$
$(1\ 0\ 1)$	2	2	_
$(1\ 1\ 1)$	3	1	$(0\ 1\ 1)$

Now, we introduce two parameters of linear block code C the (Minimum) distance of C and the minimum weight of C.

Definition 3.6: For a code C containing at least two codewords, the (minimum) distance of C, denoted by $d_{min}(C)$ or $d(C)$, is

$$d_{min}(C) = d(C) = \min\{d(x,y): x,y \in C, x \neq y\}.$$

Corollary 3.2: According to lemma 3.1, the (minimum) distance of a binary block code C is;

$$d_{min}(C) = d(C) = \min\{w(x+y): x,y \in C, x \neq y\} \quad (3.10)$$

Remark 3.3: We can denote a code $C(n,k)$ using the parameters $n, k,$ and d_{min} as (n,k,d_{min}) code, where the length of code C is n, dimension of the code C is k and d_{min} is the distance of the code C.

Example 3.6: Let $C = \{(0\,0\,0\,0\,0),(0\,0\,1\,1\,1),(1\,1\,1\,1\,1),(1\,1\,0\,0\,0)\}$ be an $(5,2)$ linear code. The minimum distance of C is $d_{min}(C) = 2$.

Since $d((0\,0\,0\,0\,0),(0\,0\,1\,1\,1)) = 3$, $d((0\,0\,0\,0\,0),(1\,1\,1\,1\,1)) = 5$,

$d\big((0\,0\,0\,0\,0),(1\,1\,0\,0\,0)\big) = 2$ $d((0\,0\,1\,1\,1),(1\,1\,1\,1\,1)) = 2$,

$d\big((0\,0\,1\,1\,1),(1\,1\,0\,0\,0)\big) = 5$, $d\big((1\,1\,1\,1\,1),(1\,1\,0\,0\,0)\big) = 3$.

Hence, C is a binary linear $(5,2,2)$ code.

Definition 3.7: The parameter $w_{min} = min\{w(c): c \in C, c \neq 0\}$ is called the minimum weight of the linear code C.

Theorem 3.3: The minimum distance of a linear block code C is equal to the minimum weight of its nonzero codewords.

Proof: $d_{min} = min\{d(x,y): x,y \in C, x \neq y\}$

$= min\{w(x+y): x,y \in C, x \neq y\}$

Since C is a linear code, so the sum of two vectors x, y is also a codeword c in $\Rightarrow \min\{w(c): c \in C, c \neq 0\} = w_{min}$. ■

In (Example 3.6) $d_{min}(C) = \min\begin{Bmatrix} w(0\,0\,1\,1\,1), w(1\,1\,1\,1\,1), \\ w(1\,1\,0\,0\,0) \end{Bmatrix}$

$= min\{3,5,2\} = 2 = w_{min}$

Next, we prove a number of theorems that relate the weight structure of a linear block code to its parity-check matrix.

Theorem 3.4: A linear block code $C(n,k)$ which has H as parity-check matrix, contains a nonzero codeword v of hamming weight l if and only if there exist l columns of H s.t. the vector sum of these l columns is equal to the zero word.

Proof:($\Longrightarrow$) Let $H = \{h_0, h_1, \ldots, h_{n-1}\}$ be the parity-check matrix for a linear code $C(n,k)$ and let $v = (v_0\ v_1\ \ldots\ v_{n-1})$ be a nonzero codeword in C s.t., $w(v) = l \Longrightarrow v$ has l nonzero components say $v_{i_1}, v_{i_2}, \ldots, v_{i_l}$ where $0 \leq i_1 < i_2 < \cdots < i_l \leq n-1$.

Now, since v is a codeword in $C \Longrightarrow 0 = H.v^T$

$$= h_0 v_0 + \cdots + h_{n-1} v_{n-1}$$
$$= h_{i_1} v_{i_1} + h_{i_2} v_{i_2} + \ldots + h_{i_l} v_{i_l}$$
$$= h_{i_1} + h_{i_2} + \cdots + h_{i_l}$$

($\Longleftarrow$) Suppose that $h_{i_1}, h_{i_2}, \ldots, h_{i_l}$ are l columns of H s.t.

$$h_{i_1} + h_{i_2} + \cdots + h_{i_l} = 0 \qquad (3.11)$$

Consider an n-tuple $x = (x_0\ \ldots\ x_{n-1})$ whose nonzero components are $x_{i_1}, x_{i_2}, \ldots, x_{i_l} \Longrightarrow w(x) = l$. We want to show that $x \in C$.

Consider the product $H.x^T = h_0 x_0 + \cdots + h_{n-1} x_{n-1}$

$= h_{i_1} x_{i_1} + h_{i_2} x_{i_2} + \ldots + h_{i_l} x_{i_l}$

$= h_{i_1} + h_{i_2} + \cdots + h_{i_l}$

$= 0$. From (3.11)

$\Longrightarrow x \in C$. ■

Corollary 3.3: Let C be a linear block code with parity-check matrix H. If no $d_{min} - 1$ or fewer columns of H add to 0, the code has minimum weight at least d_{min}.

Corollary 3.4: Let C be a linear block code with parity-check matrix H. The minimum weight or (Minimum distance) of C is equal to the smallest number of columns of H that sum to 0.

Example 3.7: Let $C = \{(0\ 0\ 0\ 0\ 0), (0\ 0\ 1\ 1\ 1), (1\ 1\ 0\ 0\ 1), (1\ 1\ 1\ 1\ 0)\}$ be $(5,2)$ linear code with the corresponding parity-check matrix

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

We see that all columns are nonzero and that no two of them are equals. Therefore, no two or fewer columns sum to 0. Hence, the minimum weight of $C(5,2)$ is at least 3.

Note that, the zeroth, first and fourth columns add up to zero, i.e.,

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

$$\Rightarrow w_{min} = d_{min} = 3.$$

3.5 Syndrome & Error Detection/ Correction

3.5.1 Syndrome & Error Detection

Consider an (n, k) linear code C. Let $v = (v_0\ v_1 \ldots\ v_{n-1})$ be a codeword that was transmitted over a noisy channel (BSC). Let $r = (r_0\ r_1\ \ldots\ r_{n-1})$ be the received vector at the output of the channel. Because of the channel noise, r may be different from v. Hence, the vector sum

$$e = r + v = (e_0\ e_1\ \ \ldots\ e_{n-1}) \tag{3.12}$$

is an n-tuple where $e_i = 1\ for\ r_i \neq v_i\ \forall\ i = 0,1, \ldots, n-1$.

This n-tuple is called an error vector or (Error pattern). The $1's$ in e are the transmission errors caused by the channel noise.

Definition 3.8: Let $C \subseteq V_n$ be an (n, k) linear code with parity-check matrix H. Then for a received word r, the syndrome of v, denoted by $s(r)$ or (s) is:

$$s(r) = s = r.H^T = (s_0 \ldots s_{n-k-1}) \tag{3.13}$$

Or $s^T = H.r^T = \begin{pmatrix} s_0 \\ \vdots \\ s_{n-k-1} \end{pmatrix}$

Note that s is a linear map $s : V_n \rightarrow V_{n-k}$

Remark 3.4:

(i) $s(r) = 0 \iff r \in C \Longrightarrow r$ is a codeword and the receiver accepts r as the transmitted codeword.

(ii) When $s \neq 0$, we know that the received word $r \notin C$ and the presence of errors has been detected.

Definition 3.9: An error pattern e is called an undetectable error pattern if it is identical to a codeword.

When a codeword v is transmitted over a noisy channel, and undetectable error pattern e occurred to the transmitted codeword v then, the received word r will be $r = v + e$, which is also a codeword since it's the sum of two codewords. Thus, the syndrome of r will be zero.

In this case, the decoder accepts r as the transmitted codeword and thus commits an incorrect decoding, and we say that the decoder makes a decoding error.

Now, let H be a parity-check matrix in a systematic form of an (n, k) linear code. Then based on (3.13), the syndrome digits are as follows;

$$s(r) = s(r_0\, r_1 \dots\, r_{n-1}) = r.H^T = (s_0 \dots s_{n-k-1})$$

$$= (r_0\, r_1 \dots\, r_{n-1}).\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \\ p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & \cdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}$$

$\Longrightarrow$ Syndrome digits are;

$$\begin{aligned} s_0 &= r_0 + r_{n-k}p_{00} + r_{n-k+1}p_{10} + \cdots + r_{n-1}p_{k-1,0} \\ s_1 &= r_1 + r_{n-k}p_{01} + r_{n-k+1}p_{11} + \cdots + r_{n-1}p_{k-1,1} \\ &\vdots \\ s_{n-k-1} &= r_{n-k-1} + r_{n-k}p_{0,n-k-1} + r_{n-k+1}p_{1,n-k-1} + .. + r_{n-1}p_{k-1,n-k-1} \end{aligned} \qquad (3.14)$$

Note that, the syndrome digits given in (3.14) can be formed by a circuit similar to an encoding circuit as follows:

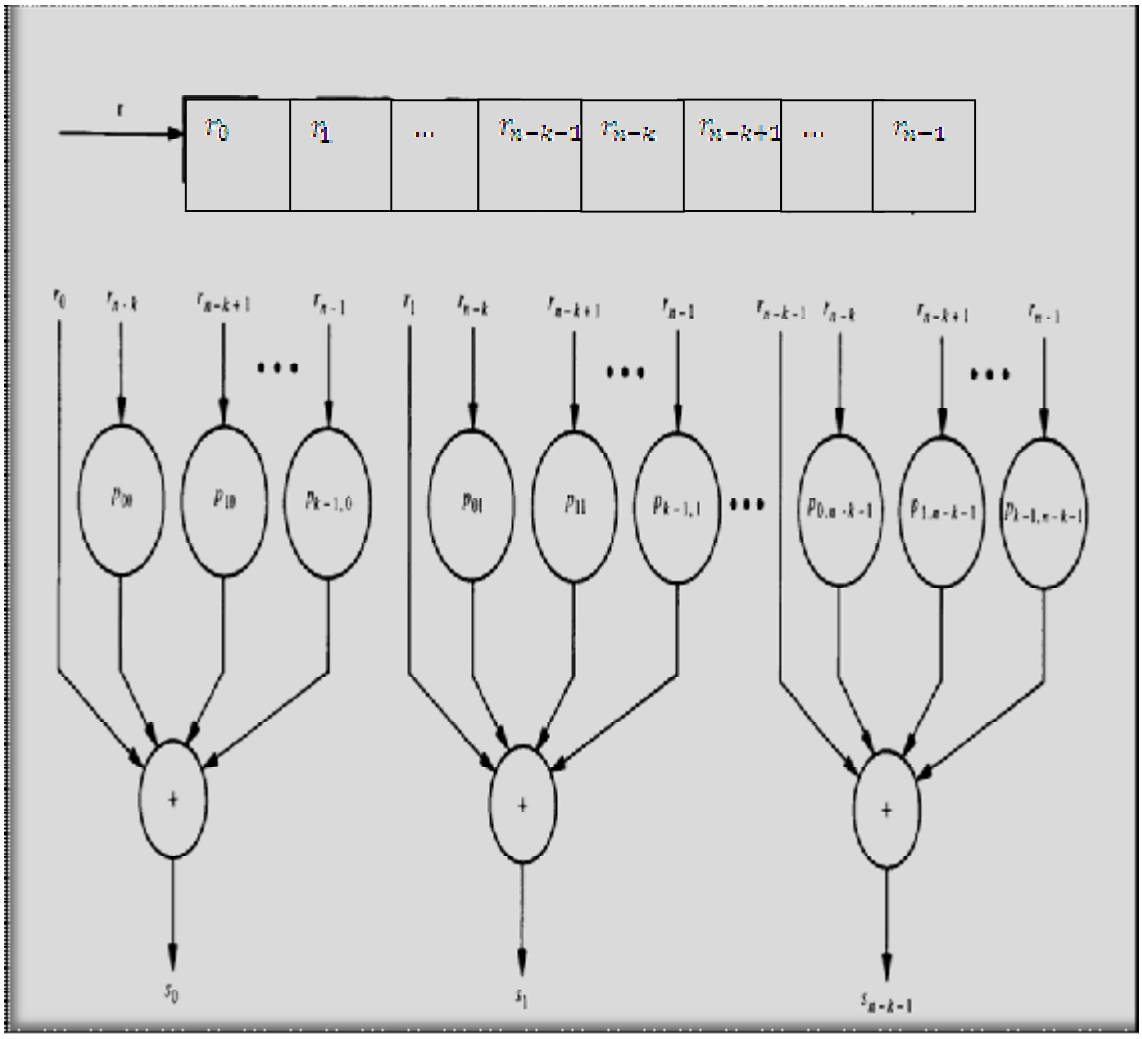


Figure 8: Syndrome Circuit for a Linear Systematic (n, k) Code

Example 3.8: Consider the $(5, 2)$ linear code whose parity-check matrix $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$ in systematic form given in example 3.7.

Let $r = (r_0\ r_1\ r_2\ r_3\ r_4)$ be the received word. Then its syndrome is given by:

$$s = (s_0\ s_1\ s_2) = (r_0\ r_1\ r_2\ r_3\ r_4).\begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\\ 1 & 1 & 1\\ 1 & 1 & 0\end{bmatrix}$$

$\Longrightarrow$ Syndrome digits are;

$$s_0 = r_0 + r_3 + r_4$$

$$s_1 = r_1 + r_3 + r_4$$

$$s_2 = r_2 + r_3$$

The syndrome circuit for this code is shown in figure 9.

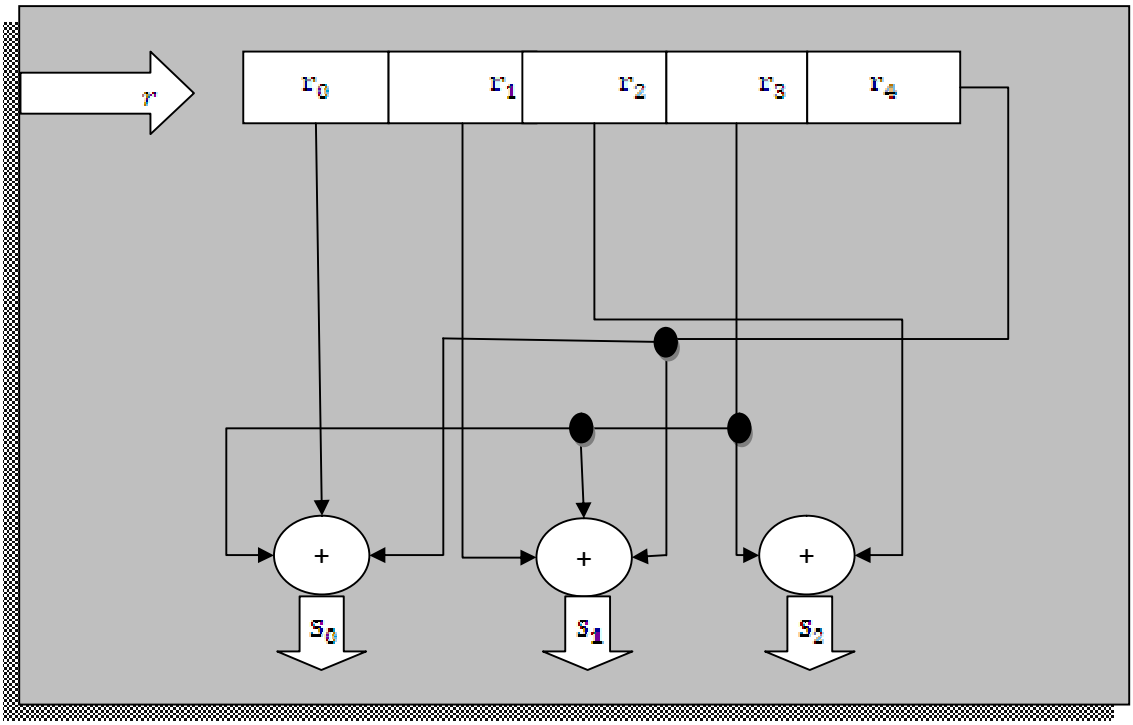


Figure 9: Syndrome Circuit for the $(5,2)$ Code

3.5.2 Syndrome & Error Correction

Theorem 3.5: The syndrome s of a received vector $r = v + e$ depends only on the error pattern e, and not on the transmitted codeword v.

Proof: Since $r = v + e$

Then by (3.13) we have;

$$s(r) = s = r.H^T = (v + e).H^T = v.H^T + e.H^T$$

But, $v.H^T = 0$ then $s(r) = e.H^T$ (3.15)

So the syndrome of r doesn't depend on v ■

We now use the syndrome for error correction; let H be a parity-check matrix in a systematic form of an (n, k) linear code C. Then the syndrome digits of the received word $r = (r_0\ r_1\ \dots\ r_{n-1})$ can be formed as follows;

$$s(r) = s(r_0\ r_1\ \dots\ r_{n-1}) = e.H^T = (s_0\ \dots\ s_{n-k-1})$$

$$= (e_0\ e_1\ \dots\ e_{n-1}).\begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 0 & \cdots & 1 \\ p_{00} & p_{01} & \cdots & p_{0,n-k-1} \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} \\ \vdots & \vdots & \cdots & \vdots \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} \end{bmatrix}$$

$$\Rightarrow s_0 = e_0 + e_{n-k}p_{00} + e_{n-k+1}p_{10} + \cdots + e_{n-1}p_{k-1,0}$$

$$s_1 = e_1 + e_{n-k}p_{01} + e_{n-k+1}p_{11} + \cdots + e_{n-1}p_{k-1,1} \quad (3.16)$$

$$\vdots$$

$$s_{n-k-1} = e_{n-k-1} + e_{n-k}p_{0,n-k-1} + e_{n-k+1}p_{1,n-k-1} + \cdots + e_{n-1}p_{k-1,n-k-1}$$

The system above (3.16) of linear equations can be solved for the digits of an error pattern $e_0, e_1, \dots, e_{n-1}$ by the following procedure for error correction which is using (3.14, 3.16).

1. Compute the syndrome $s = (s_0\ s_1\ \dots\ s_{n-k-1})$ of the received word $r = (r_0\ r_1\ \dots\ r_{n-1})$ using (3.14)
2. Solve the system of the equations in (3.16) for $e = (e_0\ e_1\ \dots\ e_{n-1})$.

Note that the system (3.16) is an $(n-k)\times n$ of linear equations, and so, it doesn't have a unique solution.

3. Compute the decoded word v^*:

$$v^* = r + e$$

Theorem 3.6: The $n-k$ linear equations of (3.16) do not have a unique solution but have 2^k solutions.

In other words, there are 2^k error patterns that result in the same syndrome, and the true error pattern e is just one of them.

Theorem 3.7: For the BSC, the most probable error pattern e is the one that has the smallest number of nonzero digits.

Example 3.9: Again, we consider the code $C(5,2)$ with the parity-check matrix $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$. Let $v = (0\ \ 0\ \ 1\ \ 1\ \ 1)$ be the transmitted codeword over BSC and $r = (1\ \ 0\ \ 1\ \ 1\ \ 1)$ be the received vector.

The problem is to find the digits of an error pattern $e = (e_0\ e_1\ e_2\ e_3\ e_4)$.

1. Compute the syndrome $s = (s_0\ s_1\ s_2)$ of $r = (1\ 0\ 1\ 1\ 1)$ using (3.14)

$$s = r.H^T = (1\ \ 0\ \ 1\ \ 1\ \ 1).\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} = (1\ \ 0\ \ 0)$$

2. Solve the system (3.16) for $e = (e_0\ e_1\ e_2\ e_3\ e_4)$ with $s = (1\ 0\ 0)$ as

$$He^T = s^T \Longrightarrow \begin{pmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}\begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} \Longrightarrow \left(\begin{array}{ccccc|c} 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ e_0 & e_1 & e_2 & e_3 & e_4 & \end{array}\right)$$

$$\Rightarrow \begin{cases} e_0 + e_3 + e_4 = 1 \\ e_1 + e_3 + e_4 = 0 \\ e_2 + e_3 = 0 \end{cases}$$

There are $2^2 = 4$ error patterns that satisfy the above system depending on $e_3\ e_4 = 0\ 0\ or\ 0\ 1\ or\ 1\ 0\ or\ 1\ 1$ they are:

$(1\ 0\ 0\ 0\ 0), (0\ 1\ 0\ 0\ 1), (0\ 1\ 1\ 1\ 0), (1\ 0\ 1\ 1\ 1)$.

Now, since the channel is BSC, Then the most probable error pattern that satisfies the system above is $e = (1\ 0\ 0\ 0\ 0)$ which has the smallest number of nonzero digits.

3. The receiver decodes the received word $r = (1\ 0\ 1\ 1\ 1)$ into the following codeword v^*;

$v^* = r + e = (1\ 0\ 1\ 1\ 1) + (1\ 0\ 0\ 0\ 0) = (0\ 0\ 1\ 1\ 1)$.

We see that the receiver has made a correct decoding.

Later we show that the $(5,2)$ linear code considered in this example is capable of correcting any single error over a span of five digits; that is, if a codeword is transmitted and if and only if one digit is changed by the channel noise, the receiver will be able to determine the true error vector and to perform a correct decoding.

3.6 Error-Detecting & Error-Correcting Capabilities of Block Codes

3.6.1 Error-Detecting Capabilities of Block Codes

Definition 3.10: Let u be a positive integer. A code C is u −error−detecting if, whenever a codeword incurs at least one and at most u errors, the resulting word is not a codeword.

Definition 3.11: A code C is exactly u −error−detecting if it is u −error−detecting but not $(u+1)$ −error−detecting.

Example 3.10: Consider the linear $C(5,2,2)$ code of Example 3.6.

$C = \{(0\ 0\ 0\ 0\ 0), (0\ 0\ 1\ 1\ 1), (1\ 1\ 1\ 1\ 1), (1\ 1\ 0\ 0\ 0)\}$. This code is a 1-error−detecting since changing any codeword in one position does not result in another codeword. In other words,

$(0\ 0\ 0\ 0\ 0) \rightarrow (0\ 0\ 1\ 1\ 1) \rightarrow$ needs to change 3-bits,

$(0\ 0\ 0\ 0\ 0) \rightarrow (1\ 1\ 1\ 1\ 1) \rightarrow$ needs to change 5-bits,

$(0\ 0\ 0\ 0\ 0) \rightarrow (1\ 1\ 0\ 0\ 0) \rightarrow$ needs to change 2-bits,

$(0\ 0\ 1\ 1\ 1) \rightarrow (1\ 1\ 1\ 1\ 1) \rightarrow$ needs to change 2-bits,

$(0\ 0\ 1\ 1\ 1) \rightarrow (1\ 1\ 0\ 0\ 0) \rightarrow$ needs to change 2-bits,

$(1\ 1\ 1\ 1\ 1) \rightarrow (1\ 1\ 0\ 0\ 0) \rightarrow$ needs to change 3-bits

In fact, C is exactly 1 −error− detecting. Since C is not a 2-error-detecting. Therefore, if two errors occur in the first and second digits of the codeword (0 0 1 1 1) we obtain the codeword (1 1 1 1 1). Hence, these two errors will not be detected.

Lemma 3.2: A block code C with minimum distance d_{min} is capable of detecting all the error patterns of $d_{min} - 1$ or fewer errors.

Proof: If the minimum distance of a block code C is d_{min}, any two distinct codewords of C differ in at least d_{min} places. For this code, no error pattern of $d_{min} - 1$ or fewer errors can change one codeword into another. Therefore, any error pattern of $d_{min} - 1$ or fewer errors will result in a received vector r that is not a codeword in C. ■

Lemma 3.3: A block code C with minimum distance d_{min} cannot detect all the error patterns of d_{min} errors.

Proof: WLOG, suppose x & y are two codewords of $C(n, k, d_{min})$ that are differ in d_{min} places.

Now, $d(x, y) = d_{min} \rightarrow w(x + y) = d_{min} \xrightarrow{C \text{ is linear code}} w(v) = d(v, 0) = d_{min}$.

Let e be an error pattern of d_{min} errors which has 1's in the corresponding positions of the 1's of v. Then $r = v + e$ will be the zero codeword.

Hence, this error will not be detected. ■

The same argument applies to error patterns of more than d_{min} errors.

Now, a cording to lemma 3.2 and lemma 3.3, we can conclude this theorem;

Theorem 3.8: A code C with minimum distance d_{min} is an exactly $(d_{min} - 1)$-error−detecting code.

Proof: Suppose C has minimum distance d_{min}. By lemma 3.2 C is capable of detecting all error patterns of $d_{min} - 1$ or fewer errors.

By lemma 3.3 C cannot detect all the error patterns of d_{min} errors. So, it is an exactly $(d_{min} - 1)$-error−detecting code. ■

For instance, Example 3.10 is an exactly **1**-error−detecting code.

Even though a linear code with minimum distance d_{min} guarantees detecting all the error patterns of $d_{min} - 1$ or fewer errors, it is also capable of detecting a large fraction of error patterns with d_{min} or more errors. For instance, consider again the $C(5,2,2)$ code given in Example 3.10. Suppose that an error pattern of 2-errors occurs during the transmission of $v = (1\ 1\ 1\ 1\ 1)$ then;

$$(1\ \mathbf{1}\ 1\ 1\ 1) \rightarrow (0\ 0\ 1\ 1\ 1) \in C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (0\ 1\ 0\ 1\ 1) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (0\ 1\ 1\ 0\ 1) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (0\ 1\ 1\ 1\ 0) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (1\ 0\ 0\ 1\ 1) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (1\ 0\ 1\ 0\ 1) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (1\ 0\ 1\ 1\ 0) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (1\ 1\ 0\ 0\ 1) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (1\ 1\ 0\ 1\ 0) \notin C$$
$$(1\ 1\ \mathbf{1}\ 1\ 1) \rightarrow (1\ 1\ 1\ 0\ 0) \notin C$$

Theorem 3.9: An (n,k) linear code is capable of detecting $2^n - 2^k$ error patterns of length n.

Proof: Among the $2^n - 1$ possible nonzero error patterns, there are $2^k - 1$ undetectable error patterns that are identical to the $2^k - 1$ nonzero codewords. Hence,

$2^n - 1 - (2^k - 1) = 2^n - 1 - 2^k + 1 = 2^n - 2^k$ detected errors ■

Remark 3.5:

(i) If an error pattern is not undetectable error pattern, the received vector will not be a codeword. Hence, the syndrome will not be zero. In this case, error will be detected.

(ii) There are exactly $2^n - 2^k$ error patterns are detectable error patterns.

(iii) For large n, 2^k is in general much smaller than 2^n. Therefore, only a small fraction of error patterns pass through the decoder without being detected.

(iv) The random-error-detecting capability of a block code with minimum distance d_{min} is $d_{min} - 1$.

3.6.2 Error-Correcting Capabilities of Block Codes

If a block code C with minimum distance d_{min} is used for random-error correction, one would like to know how many errors that the code is able to correct.

Theorem 3.10: A block code with minimum distance d_{min} guarantees correcting all the error patterns of $t = \left[\frac{d_{min}-1}{2}\right]$ or fewer errors. The parameter t is called the random-error-correcting capability of the code. ($\left[\frac{d_{min}-1}{2}\right] = t \Rightarrow t \leq \frac{d_{min}-1}{2} < t+1$) where $[\ \ ]$ denotes greatest integer function.

Proof: Let d_{min} be the minimum distance of a block code $C(n,k)$.

Since d_{min} is either odd or even let t be a positive integer s.t.

$$2t+1 \leq d_{min} \leq 2t+2 \qquad (3.17)$$

Now, let v be the transmitted codeword and r be the received word. Let x be any other codeword in C. Then, by the triangle inequality and hamming distance: $d(x,v) \leq d(x,r) + d(r,v)$ (3.18)

Suppose that an error pattern of $\dot{t}$ errors occurs during the transmission of v i.e. $d(r,v) = \dot{t}$. Since v, x are codewords in C, we have

$d(x,v) \geq d_{min}$. But $d_{min} \geq 2t+1$ (By 3.17)

Then $d(x,r) + d(r,v) \geq d(x,v) \geq d_{min} \geq 2t+1$

$\Rightarrow d(x,r) \geq 2t+1-d(r,v) = 2t+1-\dot{t}$

Case 1: If $\dot{t} = t$

$\Rightarrow\ d(x,r) \geq t+1 > t = \dot{t} = d(r,v)$

$\Rightarrow d(x,r) > d(r,v)$ (3.19)

Case 2: If $\dot{t} < t \Rightarrow t - \dot{t} > 0$ (add t in to both sides)

$$\Rightarrow 2t - \dot{t} > t \Rightarrow 2t - \dot{t} + 1 > t > \dot{t}$$

$$\Rightarrow\ d(x,r) \geq 2t+1-\dot{t} > \dot{t} = d(r,v)$$

$$\Rightarrow\ d(x,r) > d(r,v) \quad (3.20)$$

The inequality in (3.19) and (3.20) says that if an error pattern of t or fewer errors occurs, the received word r is closer by the minimum distance decoding to the transmitted codeword v than other codeword x in C. Based on the MLD this means that the conditional probability $P(r|v)$ is greater than $P(r|x)$ for $x \neq v$. So r is decoded into v, which is the actual transmitted codeword.

On the other hand, the code is not capable of correcting all the error patterns which contains more than $t = \left[\frac{d_{min}-1}{2}\right]$ errors.

i.e. there is at least one case where an error pattern contains more than t errors results in a received word r which is closer to an incorrect codeword say x than to the actual transmitted codeword v. To see this consider the following case:

Consider $C = \{(0\ 0\ 0\ 0\ 0), (1\ 1\ 0\ 0\ 1), (1\ 1\ 1\ 1\ 0), (0\ 0\ 1\ 1\ 1)\}$ be $(5,2,3)$ linear code. Choose $v = (1\ 1\ 0\ 0\ 1), x = (1\ 1\ 1\ 1\ 0) \in C(5,2,3)$ where $d(v,x) = d_{min} = 3$ and the random-error-correcting capability of the code $C(5,2,3)$ is $t = \left[\frac{d_{min}-1}{2}\right] = \left[\frac{3-1}{2}\right] = 1$.

Suppose that $v = (1\ 1\ 0\ 0\ 1)$ is transmitted and is corrupted by the error pattern $e_1 = (0\ 0\ 1\ 1\ 0)$. Then the received word is

$$r = v + e_1 = (1\ 1\ 1\ 1\ 1) \Longrightarrow e_1 = r + v$$

$$\Longrightarrow w(e_1) = w(r + v) = d(r,v) \qquad (3.21)$$

Let $x = (1\ 1\ 1\ 1\ 0)$ s.t. $x = v + e_1 + e_2$ where e_1 and e_2 be two error patterns that satisfy the following conditions:

(i) $e_1 + e_2 = v + x = (0\ 0\ 1\ 1\ 1)$

$\Longrightarrow e_2 = (0\ 0\ 1\ 1\ 1) + (0\ 0\ 1\ 1\ 0) = (0\ 0\ 0\ 0\ 1)$

(ii) e_1, e_2 don't have nonzero components in common places.

Obviously, from (i) and by using $e_1 = r + v$

$$e_2 = x + v + e_1 = x + v + (r + v) = x + r$$

$$\Longrightarrow w(e_2) = w(x + r) = d(x,r) \qquad (3.22)$$

Combining (3.21) and (3.22), we obtain the following:

$d(r,v) = 2$ and $d(x,r) = 1$.

So r must be decoded into x, not v and the error will not be corrected.

3.7 Standard Array for Linear Codes

Let us now consider another method of decoding linear block codes that uses MLD or the minimum distance decoding.

Recall that $C = \{v_1, v_2, \ldots, v_{2^k}\}$ is the set of codewords that is a subset of the set of all n-tuples $V_n = \{v_1, v_2, \ldots, v_{2^n}\}$.

In this decoding scheme V_n is evenly partition into 2^k disjoint subsets $D_1, D_2, \ldots, D_{2^k}$, s.t. each D_i contains exactly one codeword (say v_i).

Now, we build the $D_i's$ as follows:

Step 1: Start each D_i with an element v_i of C, e.g. $v_1 = (0\ 0 \ldots 0)$

D_1	D_2	...	D_i	...	D_{2^k}
$v_1 = (0\ 0 \ldots 0)$	v_2		v_i		v_{2^k}

Step 2: Choose an n-tuple $e_2 \in V_n - C$ of minimum weight. Then the second element in each D_i is $v_i + e_2$

D_1	D_2	...	D_i	...	D_{2^k}
$v_1 = (0\ 0 \ldots 0)$	v_2		v_i		v_{2^k}
$e_2 + v_1$	$e_2 + v_2$		$e_2 + v_i$		$e_2 + v_{2^k}$

Note that the element e_2 is called a coset leader.

Step 3: We repeat step 2 by choosing e_i (coset leader) where $i = 3, 4, \ldots, 2^{n-k}$ of minimum weight from the remaining n-tuples that were not included in any D_i up to the previous step

D_1	D_2	...	D_i	...	D_{2^k}
$v_1 = (0\ 0 \ldots 0)$	v_2		v_i		v_{2^k}

$$e_2 + v_1 \qquad e_2 + v_2 \qquad e_2 + v_i \qquad e_2 + v_{2^k}$$

$$\vdots$$

$$e_{2^{n-k}} + v_1 \qquad e_{2^{n-k}} + v_2 \qquad e_{2^{n-k}} + v_i \qquad e_{2^{n-k}} + v_{2^k}$$

Remark 3.6: The above n- tuples can be arranged as entries of $2^{n-k} \times 2^k$ matrix M s.t.: $m_{ij} = e_i + v_j$.

$$M_{2^{n-k}\times 2^k} = \begin{bmatrix} v_1 & v_2 & \mathbf{L} & v_i & \mathbf{L} & v_{2^k} \\ e_2 + v_1 & e_2 + v_2 & \mathbf{L} & e_2 + v_i & \mathbf{L} & e_2 + v_{2^k} \\ \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} \\ e_{2^{n-k}} + v_1 & e_{2^{n-k}} + v_2 & \mathbf{L} & e_{2^{n-k}} + v_i & \mathbf{L} & e_{2^{n-k}} + v_{2^k} \end{bmatrix}$$

Remarks 3.7:

(i) The entries of the j^{th} column of M represent the subset

$$D_j = \{v_j, e_2 + v_j, e_3 + v_j, \ldots, e_{2^{n-k}} + v_j\}. \qquad (3.23)$$

(ii) The entries of the first row of M represent the code $C = \{v_1, v_2, \ldots, v_{2^k}\}$.

(iii) The i^{th} row of M is called the i^{th} coset and the first entry of this row is a coset leader e_i.

(iv) Each n-tuple of V_n appears only once in M and hence M containing all n-tuples.

(v) The standard array built is not a unique.

(vi) Each i^{th} coset consists of 2^k n-tuples of the form

$$e_i + C = \{e_i + v_j : v_j \in C\}.$$

(vii) Each column consists of 2^{n-k} n-tuples with the topmost one as a codeword in C.

Example 3.11: Consider this linear code $C(5,2) = \{(0\ 0\ 0\ 0\ 0), (1\ 1\ 0\ 0\ 1), (1\ 1\ 1\ 1\ 0), (0\ 0\ 1\ 1\ 1)\}$. The standard array for C shown as follows:

D_1	D_2	D_3	D_4
$v_1 = 00000$	$v_2 = 11001$	$v_3 = 11110$	$v_4 = 00111$
$e_2 = 10000$	$e_2 + v_2 = 01001$	$e_2 + v_3 = 01110$	$e_2 + v_4 = 10111$
$e_3 = 01000$	$e_3 + v_2 = 10001$	$e_3 + v_3 = 10110$	$e_3 + v_4 = 01111$
$e_4 = 00100$	$e_4 + v_2 = 11101$	$e_4 + v_3 = 11010$	$e_4 + v_4 = 00011$
$e_5 = 00010$	$e_5 + v_2 = 11011$	$e_5 + v_3 = 11100$	$e_5 + v_4 = 00101$
$e_6 = 00001$	$e_6 + v_2 = 11000$	$e_6 + v_3 = 11111$	$e_6 + v_4 = 00110$
$e_7 = 10100$	$e_7 + v_2 = 01101$	$e_7 + v_3 = 01010$	$e_7 + v_4 = 10011$
$e_8 = 10010$	$e_8 + v_2 = 01011$	$e_8 + v_3 = 01100$	$e_8 + v_4 = 10101$

Figure 10: Standard Array for the $(5,2)$ Linear Code.

Theorem 3.11: The sum of any two words in the same row of the standard array is a codeword in C.

Proof: Suppose that $e_l + v_i, e_l + v_k$ be two n-tuples in the l^{th} row where $v_i, v_k \in C$ $i \neq k$ and e_l is the word with minimal weight in the l^{th} row. Then $(e_l + v_i) + (e_l + v_k) = v_i + v_k \in C$ since C is linear. ■

Theorem 3.12:

(i) No two n –tuples in the same row of a standard array are identical.

Proof: Suppose two n –tuples in the l^{th} row are identical, say $e_l + v_i = e_l + v_j$ with $i \neq j$. This means that $v_i = v_k$, which is a contradiction since 1^{st} row contains distinct codewords $v_1, v_2, \dots, v_{2^k}$. ■

(ii) Every n –tuple appears in one and only one row.

Proof: Suppose that an n-tuple v appears in both l^{th} row and the m^{th} row with $l < m$. So $v = e_l + v_i = e_m + v_j$ for some $i \neq j$.

Then $e_l + v_i = e_m + v_j \Longrightarrow e_m = e_l + (v_i + v_j)$. But v_i, v_j are two codewords in the linear code C, so $v_i + v_j$ is again a codeword in C, say v_s. Then $e_m = e_l + v_s$. This implies that the n-tuple e_m is in the l^{th} row of the array, which contradicts the construction rule of the array that e_m, the first element of the m^{th} row, should be unused in any previous row. Therefore, no n-tuple can appear in more than one row of the array ■

From the previous theorem we can conclude this corollary;

Corollary 3.5:

(i) There are 2^k disjoint columns in the standard array.

(ii) Every word appears exactly once in the standard array.

Theorem 3.13:

(i) Each row in the standard array consists of 2^k distinct elements.

Proof: Clearly, each row consists of the 2^k n-tuples since the first row containing 2^k codewords by (Theorem 3.12 (i)) no two n-tuples in the same row of a standard array are identical. ■

(ii) There are 2^{n-k} disjoint rows in the standard array.

Proof: Since there are 2^n $n-$tuples over $GF(2)$ and the are partitioned into 2^k disjoint columns so we have, $\frac{2^n}{2^k} = 2^{n-k}$ disjoint rows in the standard array. ■

Fact 3.1: Any element in a coset can be used as its coset leader. This does not change the elements of the coset; it simply permutes them.

We now, explain the standard array decoding:

Let us use the 2^k disjoint columns $D_1, D_2, \dots, D_i, \dots, D_{2^k}$ of the standard array for decoding the code C.

Suppose that the codeword v_j is transmitted over the BSC. From (3.23) if the error pattern e_i caused by channel was a coset leader, the received word r will be $r = e_i + v_j \in D_j$. So r will be decoded correctly into the transmitted codeword v_j.

On the other hand, if the error pattern caused by the channel is not a coset leader, an erroneous decoding will result as in the following example:

Let x be an error pattern caused by the channel and lie in the l^{th} coset and under the codeword $v_s \neq 0$. Then $x = e_l + v_s$ and the received word is

$r = v_j + x = e_l + (v_s + v_j) = e_l + v_r$.

Thus, the received word $r \in D_r$. So it is decoded into v_r, which is not the transmitted codeword v_j. So an erroneous decoding will result.

Result: Every (n, k) linear code is capable of correcting 2^{n-k} different error patterns which are denoted by the coset leaders. For this reason, the 2^{n-k} coset leaders (including the zero codeword) are called the correctable error patterns.

Theorem 3.14: Given an (n, k) linear code C with minimum distance d_{min}, no two n –tuples of weight $t = \left[\frac{d_{min}-1}{2}\right]$ or less can be in the same coset of C.

Proof: Let $C(n, k)$ be a linear code with minimum distance and minimum weight d_{min}. Let x and y be two n –tuples of weight t or less which are in the same coset

$w(x) \leq t$ and $w(y) \leq t$

then $x + y$ must be a nonzero codeword in C, where the weight of $x + y$ is:

$w(x + y) \leq w(x) + w(y) \leq 2t = 2\left[\frac{d_{min}-1}{2}\right] \leq 2\frac{d_{min}-1}{2} = d_{min} - 1 < d_{min}$,

But this is impossible since $w(v) \geq d_{min} \forall v \in C$. Therefore, no two n –tuples of weight t or less can be in the same coset of C. ■

Corollary 3.6: All n-tuples of weight t or less can be used as coset leaders.

Theorem 3.15: If each coset leader is chosen to have minimum weight in its coset, the decoding based on the standard array is the minimum distance decoding or the MLD.

Proof: Let r be the received word. Suppose that r is found in the i^{th} column D_i and l^{th} coset of the standard array. Then r is decoded into the codeword v_i. Since $r = e_l + v_i$, the distance between r and v_i is

$$d(r, v_i) = w(r + v_i) = w(e_l + v_i + v_i) = w(e_l) \quad (3.24)$$

Now, consider the distance between r and any codeword, say v_j,

$d(r, v_j) = w(r + v_j) = w(e_l + v_i + v_j)$,

Since v_i and v_j are two different codewords in the linear code, then the sum $v_i + v_i$ is again codeword, say v_s. Thus,

$$d(r, v_j) = w(e_l + v_s) \tag{3.25}$$

Since e_l and $e_l + v_s$ are in the same coset, then by assumption,

$$w(e_l) \leq w(e_l + v_s) \Longrightarrow d(r, v_i) \leq d(r, v_j) \text{ by (3.16 and 3.17)} \tag{3.26}$$

So the result (3.26) says that the received word is decoded into a closet codeword. ■

Theorem 3.16: If all the n-tuples of weight $t = \left[\frac{d_{min}-1}{2}\right]$ or less are used as coset leaders of the standard array for the linear code $C(n, k, d_{min})$ then there is at least one n-tuple of weight $t + 1$ that cannot be used as a coset leader.

Proof: Let v be a minimum weight codeword of C, that is $w(v) = d_{min}$. Let x and y be two n –tuples which satisfy the following two conditions:

(i) $x + y = v$,

(ii) x and y do not have nonzero components in common places.

It follows from the definition that x, y since its sum is a codeword and

$$d_{min} = w(v) = w(x + y) = w(x) + w(y).$$

Suppose we choose y s.t., $w(y) = t + 1$. Since $2t + 1 \leq d_{min} \leq 2t + 2$, we have; $2t + 1 \leq d_{min} = w(x) + w(y) = w(x) + t + 1 \leq 2t + 2$

$$\Longrightarrow t \leq w(x) \leq t + 1 \Longrightarrow w(x) = t \text{ or } t + 1,$$

Now, if x is used as a coset leader, then y cannot be a coset leader. ■

Corollary 3.7: An (n, k, d_{min}) linear code C is capable of correcting all the error patterns of t or fewer errors, but it is not capable of correcting all the error patterns of weight $t+1$.

Hence, we say that the code is exactly t −error−correcting

Example 3.12: Consider example 3.11 where the vector $v_3 = (1\ 1\ 1\ 1\ 0)$ is the transmitted codeword from $C(5,2)$ and the received word is

$r = (0\ 1\ 1\ 0\ 0)$, which lies in column D_3 whose coset leader

$e = (1\ 0\ 0\ 1\ 0)$. So e is correctable error pattern.

Other wise, consider the transmitted codeword is $v_3 = (1\ 1\ 1\ 1\ 0)$ and the received word is $r = (0\ 0\ 1\ 1\ 0)$, then the error vector is $e = v_3 + r = (1\ 1\ 0\ 0\ 0)$, which is not coset leader in the standard array for the $(5,2)$ linear code. So e is uncorrectable error pattern. To show this;

$r = (0\ 0\ 1\ 1\ 0) \xrightarrow{Coset\ leader} e = (0\ 0\ 0\ 0\ 1) \xrightarrow{r+e} v_4 = (0\ 0\ 1\ 1\ 1)$, which is not the transmitted codeword v_3. So an erroneous decoding will result.

The minimum distance for the above code is 3. Thus, the code guaranteed to correct all error patterns of single errors since the random error correcting capability of the code is

$t = \left[\frac{d_{min}-1}{2}\right] = \left[\frac{3-1}{2}\right] = 1$

and the random error detecting capability is $d_{min} - 1 = 3 - 1 = 2$

Note from the standard array of Example 3.11 not all $\binom{5}{2} = 10$ error patterns of weight two are correctable. You can select arbitrary only two of them. Because of the code C is capable of correcting $2^{n-k} = 8$ (Including

the zero word $(0\ 0\ 0\ 0\ 0)$) different coset leaders where five of them are error patterns of single errors.

Hence, the decoder in (Example 3.11) is capable of correcting all errors of weight 1;

$e_2 = (1\ 0\ 0\ 0\ 0), e_3 = (0\ 1\ 0\ 0\ 0), e_4 = (0\ 0\ 1\ 0\ 0), e_5 = (0\ 0\ 0\ 1\ 0),$

$e_6 = (0\ 0\ 0\ 0\ 1)$

and two different error patterns of weight 2;

$$e_7 = (1\ 0\ 1\ 0\ 0), e_8 = (1\ 0\ 0\ 1\ 0)$$

Theorem 3.17:

(i) All the 2^k n-tuples of a coset have the same $(n-k)$-tuple syndrome.

(ii) The syndromes for different cosets are different.

Proof: Let H be the parity check matrix of the given (n,k) linear code C.

(i) Consider the coset whose coset leader is e_l and let v_s be a word in that coset then $v_s = e_l + v_i$ for some $v_i \in C$. The syndrome of v_s is; $s(v_s) = v_s H^T = (e_l + v_i)H^T = e_l H^T + v_i H^T = e_l H^T = s(e_l)$, $(v_i H^T = 0)$. The equality above says that the syndrome of any word in a coset is equal to the syndrome of the coset leader. Therefore, all the word of a coset have the same syndrome. ■

(ii) Let e_j, e_l be the coset leaders of the j^{th}, l^{th} different cosets, respectively, where $j < l$. Suppose that the syndrome of these two cosets are equal then; $s(e_j) = s(e_l) \Rightarrow e_j H^T = e_l H^T \Rightarrow (e_j + e_l)H^T = 0 \Rightarrow (e_j + e_l) \in C$ say v_s. Thus $v_s = e_j + e_s \Longrightarrow e_l = e_j + v_i \Longrightarrow e_l$ in the j^{th} coset, thus e_l in the $l^{th}$and j^{th} cosets, which

contradicts Theorem 3.7 that states every n –tuple appears in one and only one coset in the standard array. ■

Corollary 3.8: There is a one-to-one correspondence between a coset and an $(n-k)$ tuple syndrome. Or, there is a one-to-one correspondence between a coset leader (a correctable error pattern) and a syndrome.

3.8 Syndrome Decoding

In this section we will discuss a scheme for decoding linear block codes, that uses a one-to-one correspondence between a coset leader and a syndrome. So we can form a *decoding table*, which is much simpler to use than a standard array. The table consists of 2^{n-k} coset leaders (the correctable error patterns) and their corresponding syndromes.

So the exhaustive search algorithm on the set of 2^{n-k} syndromes of correctable error patterns can be relised if we have a decoding table, in which syndromes correspond to coset leaders.

The decoding of a received word consists of three main steps:

(i) Calculation of the syndrome s of the received word r;

$$s(r) = r.H^T \Rightarrow s^T = H.r^T$$

(ii) Search the decoding-table for the coset leader e that corresponds to the syndrome s.

(iii) Decode the received word r into the codeword $v^* = r + e$.

The decoding scheme described above is called the *syndrome decoding* or *table-lookup decoding*.

Example 3.13: Consider $C(5,2,3) = \{(0\,0\,0\,0\,0),(1\,1\,0\,0\,1),(1\,1\,1\,1\,0),$ $(0\,0\,1\,1\,1)\}$ with the parity-check matrix $H = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$.

The correctable error patterns and their corresponding syndromes are shown in the following decoding table. Table 8 was constructed from standard array for the $(5,2)$ linear code given in Example 3.11.

Table 8: Decoding Table for an $(5,2)$ Linear Code

Syndrome (s)	Coset leader (e)
(0 0 0)	(0 0 0 0 0)
(1 0 0)	(1 0 0 0 0)
(0 1 0)	(0 1 0 0 0)
(0 0 1)	(0 0 1 0 0)
(1 1 1)	(0 0 0 1 0)
(1 1 0)	(0 0 0 0 1)
(1 0 1)	(1 0 1 0 0)
(0 1 1)	(1 0 0 1 0)

Suppose that the codeword $v = (1\,1\,0\,0\,1)$ is transmitted and $r = (0\,1\,0\,1\,1)$ is received. The decoding of a received word r, we use the three steps of syndrome decoding;

(i) the syndrome of r, is $s = (0\,1\,0\,1\,1).\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} = (0\,1\,1)$;

(ii) from (Table 8) the coset leader is $e = (1\,0\,0\,1\,0)$,

(iii) decode the received word r into the codeword

$$v^* = r + e = (0\,1\,0\,1\,1) + (1\,0\,0\,1\,0) = (1\,1\,0\,0\,1)$$

In that example the decoding is correct since the error pattern caused by the channel is a coset leader. Therefore, the decoding is correct if and only if

the error pattern caused by the channel is a coset leader. Otherwise we say a decoding error is committed.

3.9 Decoding Circuits Using Combinational Logic Circuits

Recall the table-lookup decoding of an (n, k) linear code from the above section. In this section the table-lookup decoding will be implemented.

The decoding table regarded as the truth table of n switching functions:

$$\begin{aligned} e_0 &= f_0(s_0, s_1, \ldots, s_{n-k-1}), \\ &\vdots \\ e_{n-1} &= f_{n-1}(s_0, s_1, \ldots, s_{n-k-1}). \end{aligned} \tag{3.27}$$

Where $s_0, s_1, \ldots, s_{n-k-1}$ are the syndrome digits, which are regarded as switching variables, and $e_0, e_1, \ldots, e_{n-k-1}$ are the estimated error digits. Now, when these n switching functions are derived and simplified, a combinational logic circuit with the $n-k$ syndrome digits as inputs and the estimated error digits as outputs can be realized.

The general decoder for an (n, k) linear code based on the table-lookup scheme is shown in Figure 11 which depends primarily on the complexity of the combinational logic circuit (error-pattern-detecting circuit).

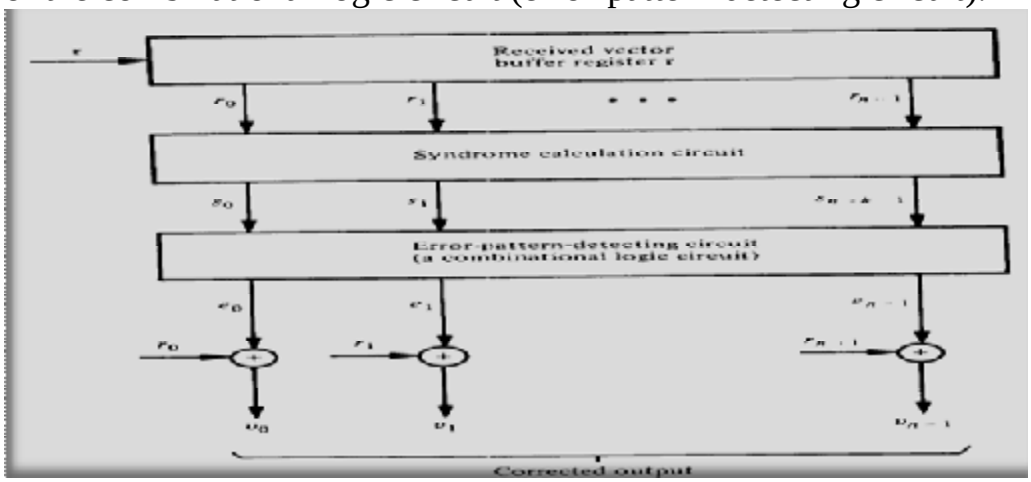


Figure 11: General Decoder for a Linear Block Code.

Example 3.12: Again consider the linear code $C(5,2)$ give in Example 3.13. The syndrome circuit for this code is shown in Figure 12. From Table 8 we form the following truth table.

Table 9: Truth Table for the Error digits of the Correctable Error patterns of the $C(5,2)$ linear code

s_0	s_1	s_2	e_0	e_1	e_2	e_3	e_4
0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0
0	1	0	0	1	0	0	0
0	0	1	0	0	1	0	0
1	1	1	0	0	0	1	0
1	1	0	0	0	0	0	1
1	0	1	1	0	1	0	0
0	1	1	1	0	0	1	0

The switching expressions for the five error digits are

$$e_0 = (s_0 \wedge s_1^{\backslash} \wedge s_2^{\backslash}) \vee (s_0 \wedge s_1^{\backslash} \wedge s_2) \vee (s_0^{\backslash} \wedge s_1 \wedge s_2),$$

$$e_1 = s_0^{\backslash} \wedge s_1 \wedge s_2^{\backslash}$$

$$e_2 = (s_0^{\backslash} \wedge s_1^{\backslash} \wedge s_2) \vee (s_0 \wedge s_1^{\backslash} \wedge s_2)$$

$$e_3 = (s_0 \wedge s_1 \wedge s_2) \vee (s_0^{\backslash} \wedge s_1 \wedge s_2)$$

$$e_4 = s_0 \wedge s_1 \wedge s_2^{\backslash}$$

Where $\wedge$ denotes the logic-AND operation, $\vee$ denotes the logic-OR operation and $s^{\backslash}$ denotes the logic-COMPLEMENT of s. These five switching expressions can be realized by seven 3-input AND gates. The circuit of this decoder is shown in Figure 12.

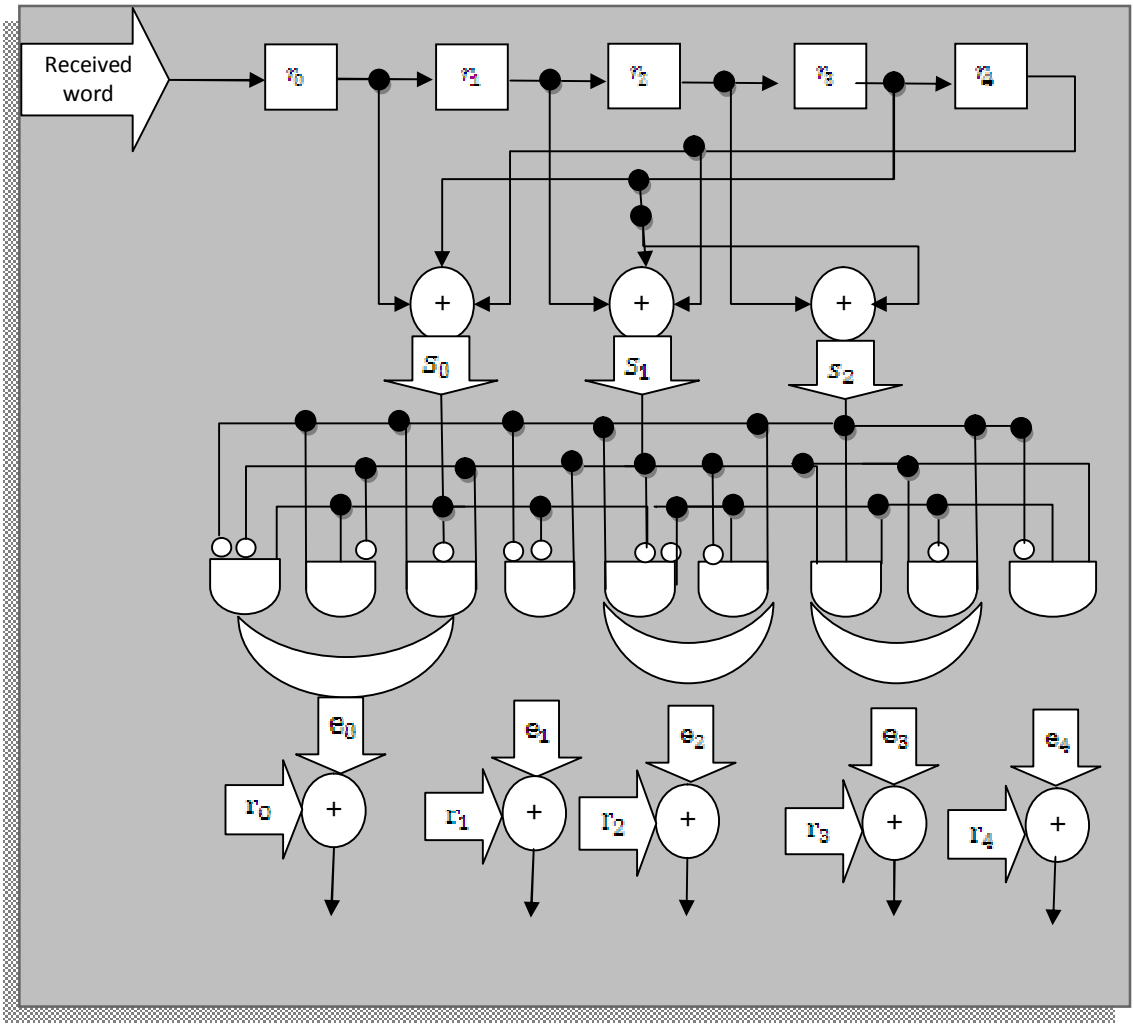


Figure 12: Decoding Circuit for the Code $\mathcal{C}(5,2)$

Chapter 4

Binary Hamming Codes

4.1 Construction of Binary Hamming Codes

Hamming codes are the first important class of linear error-correcting codes named after its inventor, Richard W. Hamming (1950) who asserted by proper encoding of information, errors induced by a noisy channel or storage medium can be reduced to any desired level without sacrificing the rate of information transmission or storage. We discuss the binary Hamming codes with their shortened and extended versions that are defined over $GF(2)$. These Hamming codes have been widely used for error control in digital communication and data storage. They have interesting properties that make encoding and decoding operations easier.

In this section we introduce Hamming codes as linear block codes that are capable of correcting any single error over the span of the code block length.

Suppose the linear code $C(n,k)$ has an $(n-k)\times n$ matrix H as the parity check matrix and that the syndrome of the received word r is given by $s^T = H.r^T$. Then the decoder must attempt to find a minimum weight e which solves the equation

$$s^T = H.e^T.$$

Write $e = (e_0, e_1, \dots, e_{n-1})$ and $H = (h_0 h_1, \dots, h_{n-1})$,

where $e_i \in GF(2)\ \forall i = 0, 1, \dots, n-1$ and each h_i is an $(n-k)$ dimensional column vector over $GF(2)$, then

$$s^T = [h_0 \ h_1 \dots h_{n-1}] \begin{pmatrix} e_0 \\ e_1 \\ \vdots \\ e_{n-1} \end{pmatrix} = \sum_{i=0}^{n-1} e_i \, h_i.$$

In other words, the syndrome may be interpreted as the vector sum of those columns of the H matrix corresponding to the positions of the errors.

Now, consider all error words of weight one are to have distinct syndromes, and then it is evidently necessary and sufficient that all columns of the H matrix must be distinct.

For if $w(e) = 1$ say $e_i = 1$ then $s_i{}^T = h_i$ if $e_j = 1$ then $s_j{}^T = h_j$ now, if $s_i{}^T \neq s_j{}^T$ then $h_i \neq h_j$ for $i \neq j$.

In other words, the parity-check matrix H of this code consists of all the nonzero $(n-k)$-tuples as its columns. Thus, there are $n = 2^{(n-k)} - 1$ possible columns.

The code resulting from above is called a Binary Hamming code of length $n = 2^m - 1$ and $k = 2^m - 1 - m$ where $m = n - k$.

Definition 4.1: For any integer $m > 1$ there is a Hamming code, Ham (m), of length $2^m - 1$ with m parity bits and $2^m - 1 - m$ information bits.

Using a binary mxn parity check matrix H whose columns are all of the m-dimensional binary vectors different from zero, the Hamming code is defined as follows:

Ham $(m) = \{v = (v_0 \, v_1 \, \dots v_{n-1}) \in V_n : H.v^T = 0\}$

Table 10: (n, k) Parameters for Some Hamming Codes

M	Hamming Code
3	(7, 4)
4	(15, 11)
5	(31, 26)
6	(63, 57)
7	(127, 120)

Theorem 4.1: The minimum distance of a Hamming code is at least 3.

Proof: If Ham (m) contained a codeword v of weight 1, then v would have 1 in the i^{th} position and zero in all other positions.

Since $Hv^T = 0 = h_i$, then i^{th} column of H must be zero. This is a contradiction of the definition of H. So Ham (m) has a minimum weight of at least 2.

If Ham (m) contained a codeword v of weight 2, then v would have 1 in the i^{th} and j^{th} positions and zero in all other positions. Again, since $H.v^T = 0 = h_i + h_j$, then h_i & h_j are not distinct. This is a contradiction. So Ham (m) has a minimum weight of at least 3.

Then $W_{min} \geq 3$. Since $d_{min} = W_{min}$ in linear codes, then $d_{min} \geq 3$, therefore the minimum distance of Hamming code is at least 3. ■

Theorem 4.2: The minimum distance of a Hamming code is exactly 3.

Proof: Let $C(n, k)$ be a Hamming code with parity-check matrix H_{mxn}. Let us express the parity-check matrix H in the following form:

$H = [h_1, .., h_i, .., h_j, .., h_{2^m-1}]$, where each h_i represents the i^{th} column of H. Since the columns of H are nonzero and distinct, no two columns add to

zero. It follows from corollary 3.3 that the minimum distance of a Hamming code is at least 3. Since H consists of all the nonzero m-tuples as its columns, the vector sum of any two columns, say h_i and h_j, must also be a column in H, say h_s i.e. $h_i + h_j = h_s$. Thus,

$h_i + h_j + h_s = 0$ (In modulo 2-addition)

It follows from (Corollary 3.4) that the minimum distance of a Hamming code is exactly 3. ■

Corollary 4.1: The Hamming code is capable of correcting all the error patterns of weight one and is capabe of detecting all 2 or fewer errors.

Proof: Use lemma 3.2 and theorem 3.10 with $d_{min} = 3$ to show this corollary as follows;

$t = \left[\frac{d_{min}-1}{2}\right] = \left[\frac{3-1}{2}\right] = 1$. So the Hamming code is capable of correcting all the error patterns of weight one.

And $d_{min} - 1 = 3 - 1 = 2$. Thus it is also has the capability of detecting all 2 or fewer errors. ■

Result For any positive integer $m \geq 1$, there exists a Hamming code $C(n, k, d_{min})$ with the following parameters:

Code length: $n = 2^m - 1$

Number of information symbols: $k = 2^m - m - 1$

Number of parity-check symbols: $n - k = m$

Random-error-correcting capability: $t = 1$ $(d_{min} = 3)$.

4.2 The Generator and the Parity Check matrices of Binary Hamming Codes Ham (m)

The Hamming code is a linear block code so we use (2.3) and (2.6) to construct the parity-check matrix H and the generator matrix G in the systematic form for this Hamming code. Rewrite (2.3) and (2.6) as follows:

$$G^* = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{k-1} \end{bmatrix} = [P \ I_k] = \begin{bmatrix} p_{00} & p_{01} & \cdots & p_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ p_{10} & p_{11} & \cdots & p_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ p_{20} & p_{21} & \cdots & p_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & \vdots & & & & \vdots & \\ p_{k-1,0} & p_{k-1,1} & \cdots & p_{k-1,n-k-1} & 1 & 0 & 0 & \cdots & 0 \end{bmatrix} \quad (4.1)$$

$$H^* = [I_{n-k} \quad P^T] = \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & \cdots & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ & & & & & \vdots & & \\ 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \quad (4.2)$$

The parameters of Hamming code are: $(n, k, d_{min}) = (2^m - 1, 2^m - 1 - m, 3)$, then the parity-check matrix H of a Hamming code is constructed by listing all non zero m-dimensional distinct columns.

Thus the systematic form of H is a matrix whose right side is all of the nonzero m-tuples of weight > 1 in any order. The left side is just the identity matrix I_m.

$$H = [I_m \ Q_{m\times k}] = \begin{bmatrix} 1 & 0 & \cdots & 0 & q_{00} & q_{10} & \cdots & q_{k-1,0} \\ 0 & 1 & \cdots & 0 & q_{01} & q_{11} & \cdots & q_{k-1,1} \\ & & & & \vdots & & & \\ 0 & 0 & \cdots & 1 & q_{0,m-1} & q_{1,m-1} & \cdots & q_{k-1,m-1} \end{bmatrix} \quad (4.3)$$

Where I_m is an $m \times m$ identity matrix and the submatrix Q consists of $k = 2^m - 1 - m$ columns which are the m –tuples of weight 2 or more.

So the generator matrix G can be obtained from H by taking the transpose of the left hand side of H with the identity I_k identity matrix on the left hand side.

$$G = \begin{bmatrix} g_0 \\ g_1 \\ g_2 \\ \vdots \\ g_{2^m-2-m} \end{bmatrix} = [Q^T \ I_k] = \tag{4.4}$$

$$\begin{bmatrix} q_{00} & q_{01} & \mathbf{K} & q_{0,m-1} & 1 & 0 & 0 & \mathbf{K} & 0 \\ q_{10} & q_{11} & \mathbf{K} & q_{1,m-1} & 0 & 1 & 0 & \mathbf{K} & 0 \\ q_{20} & q_{21} & \mathbf{K} & q_{2,m-1} & 0 & 0 & 1 & \mathbf{K} & 0 \\ \mathbf{M} & \mathbf{M} & \mathbf{K} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} \\ q_{2^m-2-m,0} & q_{2^m-2-m,1} & \mathbf{K} & q_{2^m2-m,m-1} & 1 & 0 & 0 & \mathbf{L} & 0 \end{bmatrix}$$

Note that if the parity check matrix H is not in a systematic form then by row operations or column permutations you can reorder the columns of H to obtain the systematic form of H which is resulting in an equivalent code.

Example 4.1: For $m = 4$ consider the matrix

$$H^* = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \mathbf{1} & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \mathbf{1} & 1 & 1 & 1 \end{bmatrix} \tag{4.5}$$

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

This can be considered a parity check matrix for a $(15, 11)$ Hamming code. Clearly H^* is not in a systematic form. By reordering the columns of H^* as: 1, 2, 4, 8, 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15 we obtain;

$$H = (I_4 | Q_{4x11}) = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

1 2 4 8 3 5 6 7 9 10 11 12 13 14 15

Example 4.2: For $m = 3$ there is a Hamming code $Ham(3)$, of length $n = 2^m - 1 = 2^3 - 1 = 7$, and $k = 2^m - 1 - m = 2^3 - 1 - 3 = 4$ with $\{v = (v_0\ v_1\ v_2\ v_3\ v_4\ v_5\ v_6) \in V_7 : H.v^T = 0\}$.

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

Then $H.v^T = 0 \Longrightarrow \begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} . \begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$

$$\Longrightarrow \qquad v_3 + v_4 + v_5 + v_6 = 0$$

$$v_1 + v_2 + v_5 + v_6 = 0$$

$$v_0 + v_2 + v_4 + v_6 = 0$$

The solution vector;

$v = \{v_2 + v_4 + v_6, v_2 + v_5 + v_6, v_2, v_4 + v_5 + v_6, v_4, v_5, v_6\}$. So there are $2^4 = 16$ codewords that satisfy the equations above depending on the four free variables

$v_2 v_4 v_5 v_6 = \{0000,0011,0101,0111,1001,1011,1110,0010,1111,0001,$ $1100,1010,1000,0100,0110,1101\}$.

Now $Ham\,(3) = \{(0000000), (1000011), (0100101), (0001111),$ (0011001),(0110011),(0010110),(0101010),(1111111),(1101001), (0111100),(1011010),(1110000),(1001100),(1100110),(1010101)}.

is a $(7,4)$ linear block code. One can easily check that the sum of any two codewords in this code is also a codeword.

The generator matrix G can be obtained from algorithm 1 in appendix B as follows:

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

Clearly, the matrices G and H are not in the systematic form, so the systematic form of a generator matrix G can be obtained by row operations and/or column permutations as follows; $G = \left(\begin{array}{ccc|cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right)$.

And hence, the parity-check matrix H using (4.3, 4.4) becomes;

$$H = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right),$$

which gives a different set of Hamming codewords and thus a different (7, 4) binary Hamming code. To find the code $Ham(3)$, we find the nullspace of *H*, i.,e, the set of all *7-tuples* $v = (v_0, \ldots, v_6)$ such that

$$H.v^T = 0 \Longrightarrow \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right).\begin{bmatrix} v_0 \\ v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}.$$

By computing the nullspace of *H*, we obtain

$Ham\,(3) = \{(0000000), (0010011), (0110101),$ (1000111),(1011001),(0101011),

(0011110),(1110010),(1111111), (1100001),(1101100),(1001010), (0111000),(1010100),(0100110), (0001101)}.

Clearly, H of this code consists of all nonzero distinct columns of length $m = 3$.

Now apply (Corollary 3.4) to the problem of determining d_{min}. Clearly $d_{min} \neq 1, 2$, since the columns of H are nonzero and distinct. However, there are many subsets of three columns of H, summing to 0, for example, $\begin{bmatrix}1\\0\\0\end{bmatrix} + \begin{bmatrix}0\\1\\0\end{bmatrix} + \begin{bmatrix}1\\1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\0\end{bmatrix}$. Thus $d_{min} = 3$, and so the error-correcting capability of C is $t = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor = \left\lfloor \frac{3-1}{2} \right\rfloor = 1$. That is, it is capable of correcting all error patterns of weight 1 or fewer.

4.3 Hamming Encoding

Let n be the length of the encode message. Let k be the length of message to be encoded. Therefore $n - k$ is the length of error checking digits.

Since Hamming codes are linear block codes, then the encoding operation can be described in terms of a $(2^m - m - 1) \times (2^m - 1)$ generator matrix $G = [Q^T \ I_{2^m-1-m}]$

where the codewords are obtained as linear combination of the rows of That is, the Hamming code is the row space of G.

Hence, for a message $u = (u_0 \ u_1, \ldots \ u_{2^m-m-2})$ we have the codeword $v = (v_0 \ v_1 \ldots v_{2^m-2})$ which is given by:

$$v = u.G = u.[Q^T \ I_{2^m-1-m}] = [u.Q^T \ u], \tag{4.6}$$

where $u.Q^T$ is redundant checking part of a codeword v, and u is the message part of v.

Example 4.3: Consider

$Ham\,(3) = \{(0000000), (0010011), (0110101),$

(1000111),(1011001),(0101011),(0011110),(1110010),(1111111),

(1100001),(1101100),(1001010),(0111000),(1010100),(0100110),

(0001101)}.

With the generator matrix:

$$G = \left(\begin{array}{ccc|cccc} 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 \end{array}\right).$$

Let $U = \{(0000), (1000), (0100), (1100), (0010), (1010), (0110),$

$(1110), (0001), (1001), (0101), (1101), (0011), (1011), (0111), (1111)\}$

be message set to be encoded, then its $2^k = 2^{2^m - m - 1} = 16$ corresponding codewords, according to (4.6), are shown in (Table 11) where $v = u.G$.

Table 11: A $(7,4,3)$ Hamming Code

Message (u)	Codeword (v)
(0000)	(0000000)
(0011)	(0010011)
(0101)	(0110101)
(0111)	(1000111)
(1001)	(1011001)
(1011)	(0101011)
(1110)	(0011110)
(0010)	(1110010)
(1111)	(1111111)
(0001)	(1100001)
(1100)	(1101100)
(1010)	(1001010)
(1000)	(0111000)
(0100)	(1010100)
(0110)	(0100110)
(1101)	(0001101)

Now, the weight distribution of a Hamming code of length $n = 2^m - 1$ is shown in the following definition;

Theorem 4.3: The number of codewords of weight i, A_i, of a Hamming code is simply the coefficient of z^i in the expansion of the following polynomial;

$$A(z) = \frac{1}{n+1}\{(1+z)^n + n(1-z)(1-z^2)^{\frac{n-1}{2}}\} \quad (4.7)$$

This equation is the weight enumerator for the Hamming codes.

Example 4.4: The weight enumerator for the $(7,4)$ Hamming code which is given in example 4.3 is:

$$A(z) = \frac{1}{8}\{(1+z)^7 + 7(1-z)(1-z^2)^3\} = 1 + 7z^3 + 7z^4 + z^7.$$

Hence, the weight distribution is

$A_0 = 1, A_1 = A_2 = A_5 = A_6 = 0, A_3 = A_4 = 7,$ and $A_7 = 1.$

One can easily check that distribution from table 11.

We now consider the following algorithm that is used to encode Hamming codes:

(i) All bit positions that are powers of two are used as parity bits. (Positions 1, 2, 4, 8, 16, 32, 64, etc).

(ii) All other bit positions are for the data to be encoded. (Positions 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, etc).

(iii) Each parity bit calculates the parity for some of the bits in the code word. The position of the parity bit determines the sequence of bits that it alternately checks and skips.

- Position 1 (n=1): skip 0 bit (0=n−1), check 1 bit (n), skip 1 bit (n), check 1 bit (n), skip 1 bit (n), etc. (1,3,5,7,9,11,13,15,...)
- Position 2 (n=2): skip 1 bit (1=n−1), check 2 bits (n), skip 2 bits (n), check 2 bits (n), skip 2 bits (n), etc. (2,3,6,7,10,11,14,15,...)
- Position 4 (n=4): skip 3 bits (3=n−1), check 4 bits (n), skip 4 bits (n), check 4 bits (n), skip 4 bits (n), etc. (4,5,6,7,12,13,14,15,20,21,22,…)
- Position 8 (n=8): skip 7 bits (7=n−1), check 8 bits (n), skip 8 bits (n), check 8 bits (n), skip 8 bits (n), etc. (8-15,24-31,40-47,...)
- Position 16 (n=16): skip 15 bits (15=n−1), check 16 bits (n), skip 16 bits (n), check 16 bits (n), skip 16 bits (n), etc. (16-31,48-63,...)
- Position 32 (n=32): skip 31 bits (31=n−1), check 32 bits (n), skip 32 bits (n), check 32 bits (n), skip 32 bits (n), etc. (32-63,96-127,...)
- General rule for position *n*: skip *n*−1 bits, check *n* bits, skip *n* bits, check *n* bits...
- And so on.

This general rule can be shown visually by the following table, which use d to signify data bits and p to signify parity bits.

…	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit position	
	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	P_4	d_4	d_3	d_2	P_3	d_1	P_2	P_1	Encoded data bits	
	X	X	X	X	X	X	X		X	X	X		X			Data word (without parity)	
	X		X		X		X		X		X		X		X	P1	Parity bit coverage
	X	X			X	X			X	X			X	X		P2	
	X	X	X	X					X	X	X	X				P3	
	X	X	X	X	X	X	X	X								P4	
																⋮	
	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	Data word (with parity)	

In other words, in a Hamming Code, parity bit p_i is used to hold the parity bit for all bits in the code whose locations have a binary representation with a 1 in position i.

Consider the table of four-bit binary numbers and their decimal equivalent positions shown in Table 12.

Table 12: Four-bit Numbers

$2^3 = 8$	$2^2 = 4$	$2^1 = 2$	$2^0 = 1$	Position
0	0	0	1	1
0	0	1	0	2
0	0	1	1	3
0	1	0	0	4
0	1	0	1	5
0	1	1	0	6
0	1	1	1	7
1	0	0	0	8
1	0	0	1	9
1	0	1	0	10
1	0	1	1	11
1	1	0	0	12
1	1	0	1	13
1	1	1	0	14
1	1	1	1	15

Now, since the numbers 1, 3, 5, 7, 9, 11, 13, 15, etc, have all $1's$ in position 0 in their binary representations, then p_1 is used to store the parity information for each of the bits in these locations. In other words, p_1 is used to store parity information on itself and on message bits $d_1, d_2, d_4, d_5, d_7, d_9, d_{11}$, etc.

The numbers 2, 3, 6, 7, 10, 11, 14, 15, etc, have all $1's$ in position 1 in their binary representations, then p_2 is used to store the parity information for each of the bits in these locations. In other words, p_2 is used to store parity information on itself and on message bits $d_1, d_3, d_4, d_6, d_7, d_{10}, d_{11}$, etc.

The numbers 4, 5, 6, 7, 12, 13, 14, 15, etc, have all $1's$ in position 2 in their binary representations, then p_3 is used to store the parity information for each of the bits in these locations. In other words, p_3 is used to store parity information on itself and on message bits $d_2, d_3, d_4, d_8, d_9, d_{10}, d_{11}$, etc.

And the numbers 8, 9, 10, 11, 12, 13, 14, 15, etc, have all $1's$ in position 1 in their binary representations, then p_4 is used to store the parity information for each of the bits in these locations. In other words, p_4 is used to store parity information on itself and on message bits $d_5, d_6, d_7, d_8, d_9, d_{10}, d_{11}$, etc.

After the data bits are inserted into their appropriate positions, the parity bits calculated in each case using an even parity operation which is defined in the following definition.

Definition 4.2: An even parity operation make the total number of $1's$ in a specific group of bit positions even.
For instance, since p_1 is a parity bit for the bits $d_1, d_2, d_4, d_5, d_7, d_9, d_{11}$, then if we have an odd number of 1's then $p_1 = 1$ else setting it to 0.

The calculation of an even parity can be done by applying the exclusive-or operation, denoted XOR or $\oplus$.

Table 13: Exclusive-or Operation (XOR/$\oplus$)

x y	$x \oplus y$
0 0	0
0 1	1
1 0	1
1 1	0

Remark 4.1: In the above insert the bits of a message from left to right to preserve on the binary representation of each value.

Here is an example of how this process works.

Example 4.5: For $m = 4$ consider the message

"1 0 1 0 1 1 0 1 0 1 1"

1 2 3 4 5 6 7 8 9 10 11

to be encoded then we have a $(2^m - 1 =)$ 15-bit Hamming code with 4 bits and 11 information bits.

Firstly the 11-data bits are inserted into positions:

3, 5, 6, 7, 9, 10, 11, 12, 13, 14 and 15.

"1 0 1 0 1 1 0 1 0 1 1"

And the remaining positions 1, 2, 4 and 8 (which are power of 2) are used to store parity bits which are calculated in each case using an even parity. See the table below.

Parity bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit position	
	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	P_4	d_4	d_3	d_2	P_3	d_1	P_2	P_1	Encoded data bits	
	1	1	0	1	0	1	1		0	1	0		1			Data word (without parity)	
1	1		0		0		1		0		0		1			P1	Parity bit coverage
1	1	1			0	1			0	1			1			P2	
0	1	1	0	1					0	1	0					P3	
1	1	1	0	1	0	1	1									P4	
	1	1	0	1	0	1	1	1	0	1	0	0	1	1	1	Data word (with parity)	

We write the output codeword from left to right using $p_1 = 1$ as the first bit from the left to the output codeword. Hence, the encoded codeword that would be sent is "1 1 1 0 0 1 0 1 1 1 0 1 0 1 1".

Note that, this example can be computed by placing the message-bits in the following simple table:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit position
1	1	0	1	0	1	1	؟	0	1	0	؟	1	؟	؟	Message bits

We now, after placing the data in the last table we find that in positions 3, 6, 9, 10, 12, 14 and 15 we have a "1". Using table 12 we obtain the binary representation for each of these values.

We then exclusive OR the resulting values (XOR: it sets the output to 1 if we have an odd number of 1's else setting it to 0). The results of this activity are shown below:

	8	4	2	1	positions
	0	0	1	1	3
	0	1	1	0	6
	1	0	0	1	9
	1	0	1	0	10
	1	1	0	0	12
	1	1	1	0	14
	1	1	1	1	15
XOR	$p_4 = 1$	$p_3 = 0$	$p_2 = 1$	$p_1 = 1$	

The parity bits are then put in the proper locations in the above table. Thus the following end result:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit position
1	1	0	1	0	1	1	1	0	1	0	0	1	1	1	Message bits

Thus "1 1 1 0 0 1 0 1 1 1 0 1 0 1 1" is the encoded codeword that would be sent.

4.4 Hamming Decoding

4.4.1 Syndrome & Error Detection/Correction

Suppose that a codeword v is transmitted over BSC and the received word is $r = v + e$ (e is an error pattern)

Recall that we can decode the received word r of the $C(n,k)$ linear code using property of a parity-check matrix H of that linear code which is given in (Section 2.5) as follows;

$$H.v^T = 0 \Leftrightarrow v \in C(n,k) \qquad (4.8)$$

We now use that property (4.8) of H as the decoding step, by compute the syndrome

$$s^T = H.r^T = H.(v+e)^T = H.v^T + H.e^T = H.e^T$$

Therefore, the syndrome depends only on the error pattern e and not on the transmitted codeword v.

Now, since a Hamming code is capable of correcting only a single error, suppose that e consists of zero in all positions except 1 at the i^{th} position;

$$e = (e_0 \ e_1 \ \dots \ e_i \ \dots \ e_{2^m-2}) = (0 \ \dots 0 \ 1 \ 0 \ \dots \ 0)$$

Where the 1 is equal to e_i. (That is, the error is in the i^{th} position).

Note that, an error pattern e which consists of zero in all positions except at i^{th} position can be denoted by e_i. Hence, $e_i = (0 \dots 0\ 1\ 0 \dots 0)$.

Let us express the parity-check matrix H in the following form:

$H = [h_1, .., h_i, .., h_{2^m-1}]$, where h_i represents the i^{th} column of H.

Then the syndrome is:

$$s^T = H.r^T = H.e^T = [h_1, .., h_i, .., h_{2^m-1}].\begin{bmatrix} e_0 \\ \vdots \\ e_i \\ \vdots \\ e_{2^m-1} \end{bmatrix}$$

$$= [h_1, .., h_i, .., h_{2^m-1}].\begin{bmatrix} 0 \\ \vdots \\ 0 \\ \mathbf{1} \\ 0 \\ \vdots \\ 0 \end{bmatrix} = h_i$$

$\Rightarrow s^T = h_i$, the i^{th} column of H.

Hence, the syndrome directly identifies the error location as i^{th} position. The decoding algorithm is shown as the following:

Remark 4.2: The following algorithm fails if more than one error occurs.

Algorithm:

(i) Compute the syndrome for the received binary word r, $s^T = H.r^T$,

(ii) If $s^T = 0$, then the decoded codeword is v^*, and output $v^* = r$,

(iii) Otherwise, the syndrome will be equal to a unique column of H say i denote the column of H which is equal to s^T ($s^T = h_i$). Hence, there is an error in position i of r. Add 1 (modulo-2) to the i^{th} coordinate of r, and output the result as $v^* = r + e$.

Example 4.6: Suppose that the message $u = (1\ 1\ 1\ 0)$ is encoded to the codeword $v = (0\ 0\ 1\ 1\ 1\ 1\ 0) \in Ham(3)$ which is given in example 4.3.

If the word $r = (0\ 0\ 1\ 1\ 1\ 0\ 0)$ is received, when v is transmitted over BSC, then the decoding algorithm proceeds as follows:

Consider the parity-check matrix of $Ham\ (3)$

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

Then;

(i) $s^T = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix} = H.r^T = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right).\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} = h_6,$

(ii) This syndrome corresponds to column 6 of H. Therefore, the decoded value of $r = (0\ 0\ 1\ 1\ 1\ 0\ 0)$ is;

$v^* = r + e = (0\ 0\ 1\ 1\ 1\ 0\ 0) + (0\ 0\ 0\ 0\ 0\ 1\ 0) = (0\ 0\ 1\ 1\ 1\ 1\ 0)$

Then the transmitted codeword is $v^* = (0\ 0\ 1\ 1\ 1\ 1\ 0)$, which is corresponds to the message is (1110), since the generator is systematic.

Note that the above algorithm is special case of the algorithm given in section 3.5.2. To see this;

4. $s^T = \begin{bmatrix} s_0 \\ s_1 \\ s_2 \end{bmatrix} = H.r^T = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right).\begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$

5. $$\begin{bmatrix}\mathbf{1}\\ \mathbf{1}\\ \mathbf{1}\end{bmatrix} = \left(\begin{array}{ccc|cccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{array}\right).\begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ e_4 \\ e_5 \\ e_6 \end{bmatrix}$$

$$\Rightarrow 1 = e_0 + e_4 + e_5 + e_6$$

$$1 = e_1 + e_3 + e_5 + e_6 \qquad (4.9)$$

$$1 = e_2 + e_3 + e_4 + e_5$$

Then we solve the system (4.9). The solution vector is

$$e = (1 + e_4 + e_5 + e_6, 1 + e_3 + e_5 + e_6, 1 + e_3 + e_4 + e_5, e_3, e_4, e_5, e_6)$$

.

Free variable $e_3\ e_4\ e_5\ e_6$	Error pattern $e = (e_0\ e_1\ e_2\ e_3\ e_4\ e_5\ e_6)$
0 0 0 0	(1 1 1 0 0 0 0)
0 1 0 0	(0 1 0 0 1 0 0)
0 0 1 0	(0 0 0 0 0 1 0)
0 0 0 1	(0 0 1 0 0 0 1)
1 0 0 0	(1 0 0 1 0 0 0)

But assuming the error pattern e consists of zero in all positions except at a single position, so the unique solution for that system (4.9) is (0000010). Which is the sixth column of H.

6. Compute the decoded vector v^*;

$$v^* = r + e = (0011110) + (0000010) = (0011100)$$

4.4.2 Standard Array for Hamming Codes

Recall from Theorem 3.14 for an (n, k, d_{min}) linear code C with minimum distance d_{min} all the n –tuples of weight $t = \left[\frac{d_{min}-1}{2}\right]$ or less can be used as coset leaders of a standard array of C.

So, if we form the standard array for a $(2^m - 1, 2^m - 1 - m, 3)$ Hamming code, then all the $(2^m - 1)$ –tuples of weight 1 can be used as coset leaders. Because The Hamming code is a linear single error-correcting code $(t = \left[\frac{3-1}{2}\right] = 1)$.

The number of $(2^m - 1)$-tuples of weight 1 is $2^m - 1$. Since $n - k = m$ in the Hamming codes, the code has 2^m cosets (Including the word zero).

Thus, the zero word and the $(2^m - 1)$ –tuples of weight 1 form all the coset leaders of the standard array for a $(2^m - 1, 2^m - 1 - m, 3)$ Hamming code.

This says that a Hamming code corrects only the error patterns of single error and no others.

This means that the Hamming codes belong to an extremely exclusive class of codes, the perfect codes which is defined as the following;

Definition 4.3: A t-error-correcting code is called a perfect code if its standard array has al the error patterns of t or fewer errors and no others as coset leaders.

Remark 4.3: Hamming codes form a class of single-error-correcting perfect codes. The only other binary linear perfect codes are the repetition

codes and the $(23, 12)$ Golay code (see the reference 12 for more information about these two linear perfect codes).

Example 4.7: The standard array for $Ham\ (3)$ given in table 4.2 which consists of 16 columns and 8 rows, its first row consists of $2^k = 16$ codeword, shown in Figure 13 below.

0000000	0010011	0110101	...	0100110	0001101
1000000	1010011	1110101	...	1100110	1001101
0100000	0110011	0010101	...	0000110	0101101
0010000	0000011	0100101	...	0110110	0011101
0001000	0011011	0111101	...	0101110	0000101
0000100	0010111	0110001	...	0100010	0001001
0000010	0010001	0110111	...	0100100	0001111
0000001	0010010	0110100	...	0100111	0001100

Figure 13: Standard Array for $Ham\ (3)$.

Suppose that the codeword $v = (0110101)$ is transmitted and $r = (1110101)$ is received word. For decoding r, we use figure 13 as follows;

Since r in the third column and second row of the standard array, then r is the sum of the coset leader $e_2 = (1000000)$ and the codeword $v_2 = (0110101)$. Hence,

$v^* = (1110101) + (1000000) = (0110101)$, which is the transmitted codeword.

4.4.3 Syndrome Decoding (Table-Lookup Decoding)

Decoding of Hamming codes can be accomplished easily with the table-lookup decoding described in section 3.8, which is much simpler to use than a standard array.

For instance, the standard array of the $(7,4,3)$ Hamming code ($Ham\,(3)$) given in Figure 13 can be represented by a decoding table consisting of only 2 columns instead of 16. To see this, study the following example.

Example 4.8: Consider the $(7,4)$ Hamming code given in table 11, that have the parity-check matrix

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

Then the zero word and the 7-tuples of weight 1 form all the coset leaders of the standard array for a $(7,4,3)$ Hamming code.

Thus, the correctable error patterns and their corresponding syndromes are given as follows:

Table 14: Decoding Table for a $(7,4,3)$ Hamming Code

Coset leader $e = (e_0\ e_1\ e_2\ e_3\ e_4\ e_5\ e_6)$	Syndrome $He^T = (s_0\ s_1\ s_2)$
(0000000)	(000)
(1000000)	(100)
(0100000)	(010)
(0010000)	(001)
(0001000)	(011)
(0000100)	(101)
(0000010)	(111)
(0000001)	(110)

As in Section 3.9, the table-lookup decoding of this (7, 4, 3) Hamming code may can be implemented as follow.

From the decoding table we form the truth table of seven switching functions expressions for the seven error digits. These functions are:

$$e_0 = s_0 \wedge s_1{}' \wedge s_2{}' \qquad e_1 = s_0{}' \wedge s_1 \wedge s_2{}' \qquad e_2 = s_0{}' \wedge s_1{}' \wedge s_2$$

$$e_3 = s_0{}' \wedge s_1 \wedge s_2 \qquad e_4 = s_0 \wedge s_1{}' \wedge s_2 \qquad e_5 = s_0 \wedge s_1 \wedge s_2$$

$$e_6 = s_0 \wedge s_1 \wedge s_2{}'$$

Table 15: Truth Table for the Error Digits of the Correctable Error Patterns of the (7, 4, 3) Hamming Code

Syndrome			Correctable error pattern (Coset leader)						
s_0	s_1	s_2	e_0	e_1	e_2	e_3	e_4	e_5	e_6
0	0	0	0	0	0	0	0	0	0
1	0	0	1	0	0	0	0	0	0
0	1	0	0	1	0	0	0	0	0
0	0	1	0	0	1	0	0	0	0
0	1	1	0	0	0	1	0	0	0
1	0	1	0	0	0	0	1	0	0
1	1	1	0	0	0	0	0	1	0
1	1	0	0	0	0	0	0	0	1

The complete circuit of the decoder is shown in Figure 14.

4.4.4 Checking of Parity Bits in Hamming Codes

To complete the decoding operations for $Ham\,(m)$, the received side would re-compute the parity bits and compare them to the ones received (again using an XOR – even parity). If they were the same the result will be all $0's$ i.e., no error occurred. If a single bit was flipped the location of the flipped bit is determined using Table 12.

Example 4.9 shows how this process works.

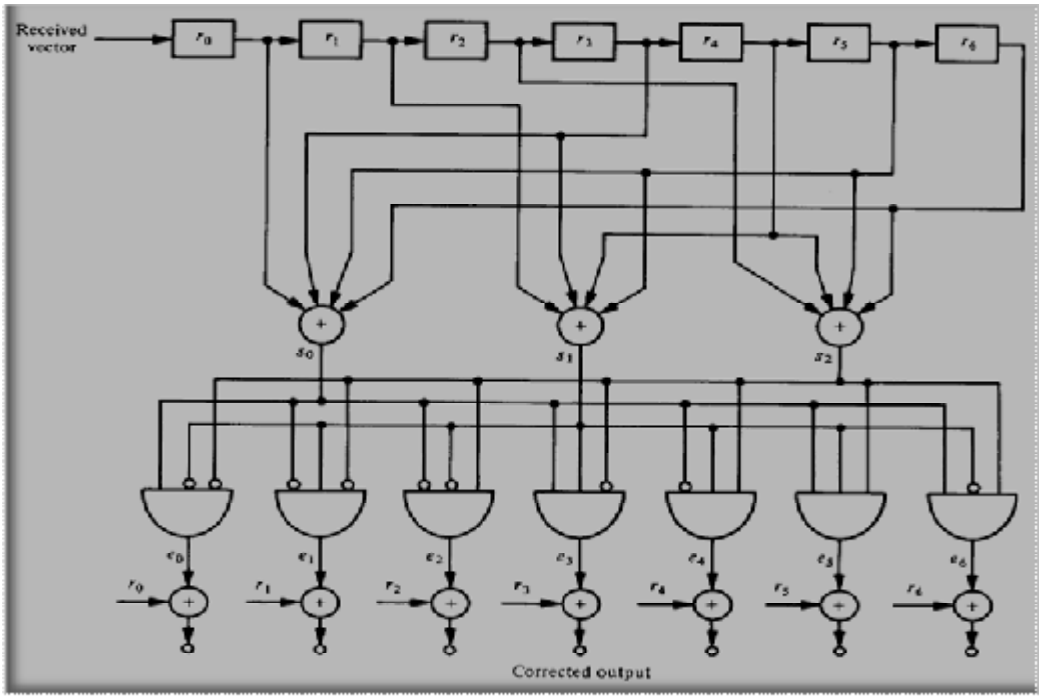


Figure 14: Decoding Circuit for the $(7,4,3)$ Code

Example 4.9: Let's say that the bit in position 14 was corrupted and turned from 1 to 0 during transmission the codeword $v = (1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1)$. The new data word (with parity bits) is now $r = (1\ 1\ 1\ 0\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1\ 0\ 0\ 1)$.

The receiving end would see the following encoded sequence:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit position
1	0	0	1	0	1	1	1	0	1	0	0	1	1	1	Received data word

Below is the re-calculation of the parity bits of $r = (111001011101001)$ using XOR – even parity in each case.

	8	4	2	1	positions
	0	0	1	1	3
	0	1	1	0	6
	1	0	0	1	9
	1	0	1	0	10
	1	1	0	0	12
	1	1	1	1	15
XOR	0	1	0	1	

The re–calculated parity information is then compared to the parity information sent/received as follows:

	8	4	2	1	parity – bit
	1	0	1	1	sent/received
	0	1	0	1	new calculated
XOR	1	1	1	0	

The final step is to evaluate the integer value of the parity bits

	p_4	p_3	p_2	p_1	Parity bit
	1	1	1	0	Binary
$\sum = 14$	$1*2^3$	$1*2^2$	$1*2^1$	$0*2^0$	decimal

Flipping the 14^{th} bit changes (1 1 1 0 0 1 0 1 1 1 0 1 0 0 1) back into (1 1 1 0 0 1 0 1 1 1 0 1 0 1 1). Removing the parity check bits gives the original data message u=(1 0 1 0 1 1 0 1 0 1 1).

All these steps can be shown as follows:

Parity bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	Bit position
	d_{11}	d_{10}	d_9	d_8	d_7	d_6	d_5	p_4	d_4	d_3	d_2	p_3	d_1	p_2	p_1	
	1	0	0	1	0	1	1	1	0	1	0	0	1	1	1	Received data word
0	1		0		0		1		0		0		1		1	p_1
1	1	0			0	1			0	1			1	1		p_2
1	1	0	0	1					0	1	0	0				p_3
1	1	0	0	1	0	1	1	1								p_4

4.5 Shortened Hamming Codes

The Hamming $(2^m - 1, 2^m - 1 - m, 3)$ can easily be shortened by deleting any l columns from the parity-check matrix H of a Hamming code. This deletion results in an $m \times (2^m - 1 - l)$ matrix H'. Now, using H' as a parity-check matrix, we obtain a shortened Hamming code with the following parameters:

Code length: $n' = 2^m - 1 - l$

Number of information symbols: $k' = 2^m - 1 - m - l$

Number of parity-check symbols: $m = n' - k'$

Minimum distance: $d_{min} \geq 3$

Theorem 4.4: The minimum distance of a shortened (n', k') Hamming code is at least 3.

Proof: Consider the parity-check matrix of a (n, k) Hamming code is a $mx(2^m - 1)$ matrix H, which consists of all the nonzero distinct columns.

If we delete any l columns from H then this deletion results in a $mx(2^m - 1 - l)$ matrix H', which is used as a parity-check matrix of a shortened Hamming code. Clearly, the matrix H' consists of nonzero and distinct columns, hence the minimum distance of a shortened (n', k') Hamming code is at least 3.

Now, we want to show that the minimum distance may be not exactly 3.

Let h_i and h_j any two columns of H', consider

$h_i + h_j = h_s$

Case 1: if $h_s \in H' \Rightarrow h_i + h_j + h_s = 0 \Rightarrow d_{min} = 3$,

Case 2: if $h_s \notin H' \Rightarrow h_i + h_j + h_s \neq 0 \quad \forall h_s \in H' \Rightarrow d_{min} \geq 3$. So the minimum distance of a shortened (n', k') Hamming code is at least 3. ■

Example 4.10:

Consider the parity-check matrix H of the $(2^m - 1, 2^m — m - 1, 3)$ Hamming code is in systematic form;

$H = (I_m | Q)$

If we delete from the submatrix Q all the columns of even weight, we obtain an $m \times 2^{m-1}$ matrix $H' = (I_m | Q')$,

where Q' consists of $2^{m-1} - m$ columns of odd weight. Since all the columns of H' have an odd weight, the sum of any two columns say h_i and h_j results in a column say h_l have even weight, that is

$h_i + h_j = h_l \notin H'$ (since H' consists of all columns of odd weight)

Thus no three columns in H' adds to zero. However, for a column h_i of weight 3 in Q', ther exists three columns h_j, h_s and h_k in I_m such that

$$h_i + h_j + h_k + h_s = 0.$$

So, the shortened Hamming code with H' as a parity-check matrix has minimum distance exactly 4 by corollary 3.4.

Theorem 4.5: The distance 4 shortened Hamming code can be used for correcting all error patterns of single error and simultaneously detecting all error patterns of triple errors or fewer.

Proof: By lemma 3.2 and theorem 3.10.

■

Decoding of the Distance 4 Shortened Hamming Code

Let $r = (r_0\ r_1\ \ldots r_{2^{m-1}-1})$ be the received word when $v = (v_0\ v_1\ \ldots v_{2^{m-1}-1})$ transmitted codeword.

If a single error occurs in the i^{th} position of v i.e.,

$$r = v + e_i = (v_0\ v_1\ \ldots v_i \ldots v_{2^{m-1}-1}) + (0\ 0\ \ldots 0\ 1\ 0 \ldots 0)$$

Thus, the syndrome of r

$s(r) = H'.e_i^T = h'_i$, where H' consists of all columns of weight odd.

The result syndrome is nonzero and it contains an odd number of 1's. However, when double errors occur, i.e.,

$r = v + e = (v_0\ v_1\ \ldots v_i \ldots v_{2^{m-1}-1}) + (0\ 0 \ldots 0\ 1\ 0 \ldots 0\ 1\ 0 \ldots 0)$

The syndrome is also nonzero, but it contains even number of 1's. Since $s(r) = H'.e^T = h'_i + h'_j$.

We now, based on these facts, decoding can be accomplished in the following manner:

(i) If $s(r) = 0$, then we assume that no error occurred,

(ii) If $s(r) \neq 0$ and it contains odd number of 1's, we assume that a single error occurred. The error pattern of a single error that corresponds to s is added to the received word for error correction.

(iii) If $s(r) \neq 0$ and it contains even number of 1's, an uncorrectable error pattern has been detected.

4.6 Extended Hamming Codes

4.6.1 Construction of Extended Hamming Codes

The Hamming (n, k, d_{min}), $Ham(m)$, can be easily extended by adding an extra parity bit to each of its codeword to obtain an $(\hat{n}, k, \hat{d}_{min})$-code called an extended Hamming code, $Ham(m)^*$

The extended code may have stronger error detection capability as we will see in the later section.

The following definition generates the extended Hamming code, $Ham(m)^*$, given a Hamming code $Ham(m)$.

We can extend the Hamming code $Ham(m)$ by extending each codeword of $Ham(m)$ by one position. This is done by adding a new parity check bit, $x_0 \in \{0, 1\}$ to the codeword, such that the weight of the codeword is even. Then the resulting is an extended Hamming code $Ham(m)^*$.

This new parity check bit, x_0 is sometimes called a parity check digit.

For instance, if $v = (v_1\ v_2 \ldots\ v_n) \in Ham(m)$

$\Longrightarrow v^* = (x_0\ v_1\ v_2\ \ldots v_n) \in Ham(m)^* \iff w(v^*)$ is even.

Example 4.11: $Ham(3)^*$ based on the Hamming code $Ham(3)$, given in (Table 11) is shown in the following table.

Table 16: $Ham(3)^*$

$Ham(3)$	$Ham(3)^*$
(0000000)	(00000000)
(0010011)	(10010011)
(0110101)	(00110101)
(1000111)	(01000111)
(1011001)	(01011001)
(0101011)	(00101011)
(0011110)	(00011110)
(1110010)	(01110010)
(1111111)	(11111111)
(1100001)	(11100001)
(1101100)	(01101100)
(1001010)	(11001010)
(0111000)	(10111000)
(1010100)	(11010100)
(0100110)	(10100110)
(0001101)	(10001101)

Theorem 4.6: An extended Hamming code $Ham(m)^*$ of an $(n, k, 3)$ Hamming code, $Ham(m)$, is an $(\acute{n}, \acute{k}, 4)$ linear code.

Where $n = 2^m - 1, k = 2^m - 1 - m$, $\acute{n} = n + 1, \acute{k} = k$ & the numbers 3 and 4 are the minimum distance of $Ham(3)$ and $Ham(3)^*$, respectively.

Proof:

From the definition of extended hamming code the parity check will be added by one bit, so the new length is $n + 1$.

The length of message doesn't change because the new bit is added as a parity check not information bit, so the number of information bits still k.

In order to show that $Ham(m)^*$ is a linear, Let $v_1{}^*$ and $v_2{}^* \in Ham(m)^*$ s.t., v_1 and v_2 are the corresponding codewords of $Ham(m)$. The vector $v_1{}^* + v_2{}^*$ will be identical to $v_1 + v_2$ in the last $2^m - 1$ positions.

If $v_1 + v_2$ has odd weight then we can, without loss of generality, assume that v_1 has odd weight and v_2 has even weight. Thus the first entry of $v_1{}^*$ is 1 and the first entry of $v_2{}^*$ is 0. It follows that the first entry of $v_1{}^* + v_2{}^*$ must be 1. Therefore $w(v_1{}^* + v_2{}^*)$ is even.

Since $v_1 + v_2 \in Ham(m)$, then $v_1{}^* + v_2{}^* \in Ham(m)^*$.

If $v_1 + v_2$ has even weight, then v_1 and v_2 must be either even or both odd. In either case, their first positions have the same entry which is 0, because of 0 + 0 = 0 and 1 + 1 = 0. Therefore $w(v_1{}^* + v_2{}^*)$ is even.

Since $v_1 + v_2 \in Ham(m)$, then $v_1{}^* + v_2{}^* \in Ham(m)^*$.

We now show that the minimum distance of $Ham(m)^*$ is 4.

Since minimum weight of $Ham(m)$ is 3, suppose $d(v_1, v_2) = 3$ for some Hamming codewords v_1 & v_2. Then one of v_1 or v_2 has even parity and the other has parity odd, say v_1 has even parity. Suppose $v_1{}^*$ and $v_2{}^*$ are the extended Hamming codewords obtained by adding check digits x_0, y_0 to v_1 and v_2, respectively. Then $x_0 = 0$ since v_1 has even parity and $y_0 = 1$ since y has odd parity.

So, $d(v_1{}^*, v_2{}^*) = d(v_1, v_2) + d(x_0, y_0) = 3 + 1 = 4$. ■

Definition 4.4: The parity check H^* of $Ham(m)^*$ is obtained by the parity check matrix H of $Ham(m)$ by adding a zero column on the left, and a row of all 1's on the bottom as follow:

$$H^*_{(m+1)\times(n+1)} = \left(\begin{array}{c|c} 0 & \\ \vdots & H_{m\times n} \\ 0 & \\ \hline 1 & 1 \ \ldots \ 1 \end{array}\right)$$

4.6.2 Error-Detecting & Error-Correcting Capabilities of Extended Hamming Codes

Corollary 4.2: For $Ham(m)^*$, we can detect up to 3 errors but still can only correct a single error.

Proof: Since the distance of $Ham(m)^*$ is 4, then we have the following;

The random-error-detecting capability of $Ham(m)^*$=$d_{min} - 1 = 3$

The random-error-correcting capability of $Ham(m)^*$=$t = \left[\frac{d_{min}-1}{2}\right] = 1$. ■

Let $H = (h_1 \ldots h_n)$ be check matrix of $Ham(m)$ and $H^* = \left(\begin{array}{c|c} 0 & \\ \vdots & H \\ 0 & \\ \hline 1 & 1 \ \ldots \ 1 \end{array}\right)$ be check matrix for an extended Hamming code, $Ham(m)^*$.

If $v^* = (x_0\ v_1 \ldots\ v_n) \in Ham(m)^*$ be transmitted codeword and $r^* = (x_0\ r_1\ \ \ldots\ \ r_n)$ be a received word with only one error. Then using the syndrome we can detect/correct that error.

First, if the error occurred in the last n bits.

Computing the syndrome of r^* where the first column of H^*, $\begin{bmatrix}0\\ \vdots \\ 0\\ 1\end{bmatrix}$ denotes by $\begin{bmatrix}h_0\\ 1\end{bmatrix}$.

$$s(r^*) = H^*.(r^*)^T = \left(\begin{array}{c|c} 0 & \\ \vdots & H \\ 0 & \\ \hline 1 & \ldots \ 1 \end{array}\right).\begin{bmatrix}x_0\\ r_1\\ \vdots\\ r_n\end{bmatrix} = \left(\begin{array}{c|c} h_0 & h_1 \ \ldots \ h_n \\ \hline 1 & 1 \ \ldots \ 1 \end{array}\right).\begin{bmatrix}x_0\\ r_1\\ \vdots\\ r_n\end{bmatrix}$$

$$= x_0\begin{bmatrix}h_0\\ 1\end{bmatrix} + r_1\begin{bmatrix}h_1\\ 1\end{bmatrix} + \cdots + r_n\begin{bmatrix}h_n\\ 1\end{bmatrix}$$

$$= \begin{bmatrix}x_0h_0\\ x_0\end{bmatrix} + \begin{bmatrix}r_1h_1\\ r_1\end{bmatrix} + \cdots + \begin{bmatrix}r_nh_n\\ r_n\end{bmatrix}$$

$$- \begin{bmatrix}x_0h_0 + r_1h_1 + \cdots + r_nh_n\\ x_0 \ + \ r_1 + \ \ldots \ \ r_n\end{bmatrix}$$

$$- \begin{bmatrix}0 + r_1h_1 + \cdots + r_nh_n\\ x_0 + r_1 + \cdots + \ r_n\end{bmatrix}$$

$$= \begin{bmatrix}r_1h_1 + \cdots + \ r_nh_n\\ x_0 \ | \ r_1 \ | \ \cdots \ | \ r_n\end{bmatrix} \qquad (4.10)$$

But the syndrome of the corresponding codeword of r^*, say r is;

$$s(r) = H.r^T = (h_1 \ldots h_n).\begin{bmatrix}r_1\\ \vdots\\ r_n\end{bmatrix} = [r_1h_1 + \cdots r_nh_n] = h_i \neq 0$$

Since there is an error in one of bits of r.

So (4.10) will be as follows;

$$s(r^*) = \begin{bmatrix}s(r)\\ x_0 + r_1 + \cdots + r_n\end{bmatrix}$$

The last row of the syndrome $s(r^*)$ will be 1, i.e. $x_0 + r_1 + \cdots + r_n = 1$

Since $w(r^*)$ is an even number, but there is an error in one bit of r^*. Hence, $w(r^*)$ will be an odd number.

So, $s(r^*) = \begin{bmatrix}h_i\\ 1\end{bmatrix} = h_i{}^*$ (4.11)

Clearly from (4.11) that the syndrome matches a column of H^*.

Hence, there is an error in position i of r^*. Add 1 (modulo-2) to the i^{th} coordinate of r^*.

Second, Suppose now that only the parity bit is in error. Then $s(r)$ will be zero, so $s(r^*) = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix}$ and this matches the first column of H^*.

Thus, we can assume 1 error has occurred and switch the bit of the word corresponding to that column.

Now suppose 2 errors have occurred. Wherever they occur the parity of the entire word will be correct, thus the syndrome will have a 0 in the last row and will not be a column of the check matrix. But the syndrome will not be zero since codewords of the extended Hamming code have minimum distance 4. Thus a nonzero, non-column syndrome indicates 2 errors.

Example 4.12: Consider the (7, 4) Hamming code given in table 11, with the check matrix $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 \end{bmatrix} = [h_1\, h_2\, h_3\, h_4\, h_5\, h_6\, h_7]$.

Then $H^* = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} = [h_0{}^*\, h_1{}^*\, h_2{}^*\, h_3{}^*\, h_4{}^*\, h_5{}^*\, h_6{}^*\, h_7{}^*]$

is the parity check matrix of an extended Hamming code, $Ham(3)^*$.

If $r = (0\,1\,1\,0\,0\,1\,1\,1)$ be a received word then:

$$s^T(r) = H^*.r^T = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} h_2 \\ 1 \end{bmatrix} = h_2{}^*$$

Hence, there is an error in position 2 of r. Add 1 to the 2^{th} coordinate of r so, $v = (0\,1\,0\,0\,0\,1\,1\,1)$.

If $r = (1\,1\,0\,0\,0\,1\,1\,1)$ be a received word then:

$$s^T(r) = H^*.r^T = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = h_0{}^*$$

Hence, there is an error in position 0 of r. Add 1 to the 0^{th} coordinate of r so, $v = (0\ 1\ 0\ 0\ 0\ 1\ 1\ 1)$.

If $r = (1\ 1\ 1\ 0\ 0\ 1\ 1\ 1)$ be a received word then:

$$s^T(r) = H^*.r^T = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

Thus the syndrome will have a 0 in the last row and will not be a column of the check matrix. Thus a nonzero, non-column syndrome indicates 2 errors.

Chapter 5

Cyclic codes

Cyclic codes form an important subclass of linear block codes and were first studied by Prange in 1957. These codes are popular for two main reasons: first, they are very effective for error detection/correction and second, they possess many algebraic properties that simplify the encoding and the decoding implementations.

5.1 Description of Cyclic Codes

If the components of an n-tuple $v = (v_0\ v_1 \dots v_{n-1})$ are cyclically shifted one place to the right, we obtain another n-tuple,

$$v^{(1)} = (v_{n-1}\ v_0 \dots v_{n-2}),$$

which is called a cyclic shift *of* v.

Clearly, the cyclic shift of v is obtained by moving the right most digit v_{n-1} of v to the left most digit and moving every other digit $v_0, v_1, \dots, v_{n-2}$ one position to the right.

If the components of v are cyclically shifted i places to the right, the resultant n-tuple would be

$$v^{(i)} = (v_{n-i}\ v_{n-i+1} \dots v_{n-1}\ v_0\ v_1 \dots v_{n-i-1}). \qquad (5.1)$$

Remark 5.1: Cyclically shifting v i-places to the right is equivalent to cyclically shifting v $(n-i)$-places to the left.

Definition 5.1: An (n, k) linear code C is called cyclic if any cyclic shift of a codeword in C is also a codeword in C, i.e. whenever $(v_0\ v_1 \dots v_{n-1}) \in C$, then so is $(v_{n-1}\ v_0 \dots\ v_{n-2})$.

Example 5.1: Consider the following $(7,4)$ linear code C;

$C = \{(0000000), (1101000), (0110100), (1011100), (0011010),$
$(1110010), (0101110), (1000110), (0001101), (1100101), (0111001),$
$(1010001), (0010111), (1111111), (0100011), (1001011)\}$.

One can easily check that the cyclic shift of a codeword in C is also a codeword in C. For instance, let $v = (1101000) \in C$, then: $v^{(1)} = (0110100) \in C$.

Hence, the code C is a cyclic.

Remark 5.2: The pair $(n, k) = (7,4)$ is not arbitrary chosen.

We show later in section (5.3) that $n - k$ is the degree of the polynomial that generates the cyclic code $C(n, k)$.

5.2 Algebraic Property of Cyclic Codes

In this section we prove an important algebraic property of cyclic codes which simplifies the encoding and syndrome computation.

Recall the polynomial representation

$$v(x) = v_0 + v_1 x + v_2 x^2 + \cdots + v_{n-1} x^{n-1} \qquad (5.2)$$

of the n-tuple $v = (v_0\ v_1 \dots v_{n-1})$ as defined in Section 1.8..

Each codeword corresponds to a polynomial of degree $\leq n - 1$.

We shall call the polynomial $v(x)$ the code polynomial *of* v.

Remark 5.3: The correspondence between the codeword v and the polynomial $v(x)$ is one-to-one. So from now on, we will use the terms "codeword" and "code polynomial" interchangeably.

Example 5.2: The polynomial representation of the $(7,4)$ cyclic code $\mathcal{C}$ of Example 5.1 is given in following table:

Table 17: The Polynomial Representation of the $(7,4)$ Cyclic Code

Codeword (v)	Code polynomial $v(x)$
v_0 =(0000000)	0
v_1 =(1101000)	$1+x+x^3$
v_2 =(0110100)	$x+x^2+x^4$
v_3 =(1011100)	$1+x^2+x^3+x^4$
v_4 =(0011010)	$x^2+x^3+x^5$
v_5 =(1110010)	$1+x+x^2+x^5$
v_6 =(0101110)	$x+x^3+x^4+x^5$
v_7 =(1000110)	$1+x^4+x^5$
v_8 =(0001101)	$x^3+x^4+x^6$
v_9 =(1100101)	$1+x+x^4+x^6$
v_{10} =(0111001)	$x+x^2+x^3+x^6$
v_{11} =(1010001)	$1+x^2+x^6$
v_{12} =(0010111)	$x^2+x^4+x^5+x^6$
v_{13} =(1111111)	$1+x+x^2+x^3+x^4+x^5+x^6$
v_{14} =(0100011)	$x+x^5+x^6$
v_{15} =(1001011)	$1+x^3+x^5+x^6$

Clearly, each codeword in (Table 17) corresponds to a polynomial of degree ≤ 6. The nonzero code polynomial of minimum degree in this cyclic code is $1+x+x^3$ and it's of degree 3.

The code polynomial that corresponds to the codeword $v^{(i)} = (v_{n-i}\ v_{n-i+1}\ \dots\ v_{n-1}\ v_0\ v_1\ \dots\ v_{n-i-1})$ is

$$v^{(i)}(x) = v_{n-i} + v_{n-i+1}x + \cdots + v_{n-1}x^{i-1} + v_0x^i + v_1x^{i+1} + \cdots + v_{n-i-1}x^{n-1} \qquad (5.3)$$

The following theorem shows an interesting algebraic relationship between $v(x)$ and $v^{(i)}(x)$.

Theorem 5.1: In the cyclic $C(n,k)$ code, the code polynomial $v^{(i)}(x)$ is simply the remainder resulting from dividing the polynomial $x^i.v(x)$ by x^n+1.

Proof: Consider the codeword $v=(v_0\ v_1 \dots v_i \dots v_{n-i} \dots v_{n-1})$ in the cyclic code $C(n,k)$.

The code polynomial that corresponds to that codeword v is

$$v(x)=v_0+v_1x+\cdots+v_ix^i+\cdots+v_{n-i}x^{n-i}+\cdots+v_{n-1}x^{n-1}.$$

Now, if the components of that v are cyclically shifted i-places to the right, then the corresponding codeword will be

$$v^{(i)}=(v_{n-i}\ \ v_{n-i+1}\ \cdots\ v_{n-1}\ \ v_0\ \ v_1\ \cdots\ v_i\ \cdots\ v_{n-i-1}).$$

The code polynomial that corresponds to the code word $v^{(i)}$ is given in (5.3).

Now, multiplying $v(x)$ by x^i, we obtain

$$\begin{aligned} x^i.v(x) &= v_0x^i+v_1x^{i+1}+\cdots+v_ix^{2i}+\cdots+v_{n-1}x^{n+i-1}. \\ &= v_0x^i+\cdots+v_{n-i-1}x^{n-1}+x^n\left(v_{n-i}+\cdots+v_{n-1}x^{i-1}\right) \\ &= v_0x^i+\cdots+v_ix^{2i}+\cdots+v_{n-i-1}x^{n-1}+x^n.q_i(x) \qquad (5.4) \end{aligned}$$

where $q_i(x)=v_{n-i}+v_{n-i+1}x+\cdots+v_{n-1}x^{i-1}.$ (5.5)

Since $q(x)+q(x)\equiv 0$ modulo 2-addition and by (5.3),

$$\begin{aligned} x^i.v(x) &= x^n.q_i(x)+q_i(x)+(q_i(x)+v_0x^i+v_1x^{i+1}\dots+v_{n-i-1}x^{n-1}) \\ &= q_i(x)(x^n+1)+v^{(i)}(x) \qquad (5.6) \end{aligned}$$

Then the code polynomial $v^{(i)}(x)$ is simply the remainder resulting from dividing the polynomial $x^i.v(x)$ by $x^n + 1$. ■

5.3 The Generator Polynomial and its Algebraic Properties

In this section, we prove number of important algebraic properties of the polynomial called the generator of the code.

Property 1: The nonzero code polynomial of minimum degree in a cyclic code C is unique.

Proof: Suppose that $g(x) = g_0 + g_1x + \cdots + g_{r-1}x^{r-1} + x^r$ is a non zero code polynomial of minimum degree r.

Suppose $g(x)$ is not unique.

Then there exists another code polynomial of degree r, say $g^*(x) = g_0^* + g_1^*x + \cdots + g_{r-1}^*x^{r-1} + x^r$.

Since C is linear, $g(x) + g^*(x) = (g_0 + g_0^*) + (g_1 + g_1^*)x + \cdots + (g_{r-1} + g_{r-1}^*)x^{r-1}$ is also a code polynomial which has degree less than r.

If $g(x) + g^*(x) \neq 0$, then $g(x) + g^*(x)$ is a nonzero code polynomial with degree less than the minimum degree r, a contradiction to the minimally of r. Thus $g(x) + g^*(x) = 0$. This implies that $g^*(x) = g(x)$. Hence, $g(x)$ is unique. ■

From now on, we use $g(x) = g_0 + g_1x + \cdots + g_{r-1}x^{r-1} + x^r$ as the unique non zero polynomial of minimum degree in C.

Property 2: The constant term g_0 must be equal to 1.

Proof: Suppose that $g_0 = 0$. Then $g(x) = g_1 x + .. + g_{r-1}x^{r-1} + x^r \Longrightarrow v = (0\ g_1 \dots g_{r-1}\ 1\ 0 \dots 0)$.

By shifting $g(x)$ cyclically $n-1$ places to the right (or one place to the left), we obtain nonzero code polynomial,

$g^{(n-1)}(x) = g_1 + g_2 x + .. g_{r-1}x^{r-2} + x^{r-1} \Longrightarrow v =$
$(g_1\ g_2 \dots g_{r-1} 1\ 0 \dots 0)$

which has a degree $r-1 < r$, a contradiction to the minimally of r. Thus $g_0 \neq 0$. ■

Example 5.3: Consider the $(7,4)$ cyclic code given in table 17. The unique nonzero code polynomial with minimum degree is:

$$g(x) = 1 + x + x^3.$$

Moreover, we treat the coefficients of $g(x)$ as the components of a codeword $v = (1\ 1\ 0\ 1\ 0\ 0\ 0)$.

Theorem 5.2: The cyclic $(n-r)$-shifts of the minimal degree code polynomial $g(x) = 1 + g_1 x + .. + g_{r-1}x^{r-1} + x^r$, given by;

$$1. g(x), xg(x), x^2 g(x), \dots, x^{n-r-1} g(x)$$

(5.7)

are all code polynomials in C. Moreover, all linear combinations of (5.7) are also code polynomials.

Proof: Consider the polynomials $xg(x), x^2 g(x), \dots, x^{n-r-1} g(x)$, which have degrees $r+1, r+2, \dots, n-1$, respectively.

Since $g_{r+1} = g_{r+2} = \cdots = g_{n-1} = 0$, then by replacing $v(x)$ by $g(x)$ in (5.5 and 5.6) we obtain the following:

$$xg(x) = g_{n-1}(x^n + 1) + g^{(1)}(x) = g^{(1)}(x)$$

$$x^2 g(x) = (g_{n-2} + g_{n-1}x)(x^n + 1) + g^{(2)}(x) = g^{(2)}(x)$$

$$\vdots \qquad (5.8)$$

$$x^{n-r-1}g(x) = g_{r+1} + g_{r+2}x + .. + g_{n-1}x^{n-r-2}(x^n + 1) + g^{(n-r-1)}(x)$$
$$= g^{(n-r-1)}(x)$$

That is, (5.8) are cyclic shifts of the code polynomial $g(x)$ in an (n, k) cyclic code C.

Using definition 5.1 of the cyclic code, then (5.7) are also code polynomials in C.

Now, since C is linear code, then all linear combinations of (5.7),

$$v(x) = u_0 g(x) + u_1 xg(x) + \cdots + u_{n-r-1}x^{n-r-1}g(x)$$

are also code polynomials where $u_i = 0$ or 1. ■

Example 5.4: Consider the cyclic $C(7,4)$ code given in table 17 with $g(x) = 1 + x + x^3$ where the corresponding codeword is $v_1 = (1\ 1\ 0\ 1\ 0\ 0\ 0)$. Then the following polynomials are cyclic shifts of $g(x)$ and also code polynomials in $C(7,4)$;

$$x.g(x) = x + x^2 + x^4 \Rightarrow v_2 = (0\ 1\ 1\ 0\ 1\ 0\ 0) \in C$$

$$x^2.g(x) = x^2 + x^3 + x^5 \Rightarrow v_4 = (0\ 0\ 1\ 1\ 0\ 1\ 0) \in C$$

$$x^{n-r-1}.g(x) = x^{7-3-1}.g(x) = x^3.g(x) =$$

$$x^3 + x^4 + x^6 \Rightarrow v_8 = (0\ 0\ 0\ 1\ 1\ 0\ 1) \in C$$

Moreover, all linear combinations of $g(x), xg(x), x^2g(x), x^3g(x)$ are code polynomials in $C(7,4)$;

$$g(x) + xg(x) = 1 + x^2 + x^3 + x^4 \Rightarrow v_3 = (1\,0\,1\,1\,1\,0\,0)$$

$$g(x) + x^2g(x) = 1 + x + x^2 + x^5 \Rightarrow v_5 = (1\,1\,1\,0\,0\,1\,0)$$

$$g(x) + x^3g(x) = 1 + x + x^4 + x^6 \Rightarrow v_9 = (1\,1\,0\,0\,1\,0\,1)$$

$$xg(x) + x^2g(x) = x + x^3 + x^4 + x^5 \Rightarrow v_6 = (0\,1\,0\,1\,1\,1\,0)$$

$$xg(x) + x^3g(x) = x + x^2 + x^3 + x^6 \Rightarrow v_{10} = (0\,1\,1\,1\,0\,0\,1)$$

$$x^2g(x) + x^3g(x) = x^2 + x^4 + x^5 + x^6 \Rightarrow v_{12} = (0\,0\,1\,0\,1\,1\,1)$$

$$g(x) + xg(x) + x^2g(x) = 1 + x^4 + x^5 \Rightarrow v_7 = (1\,0\,0\,0\,1\,1\,0)$$

$$g(x) + xg(x) + x^3g(x) = 1 + x^2 + x^6 \Rightarrow v_{11} = (1\,0\,1\,0\,0\,0\,1)$$

$$g(x) + x^2g(x) + x^3g(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$$

$$\Rightarrow v_{13} = (1\,1\,1\,1\,1\,1\,1)$$

$$xg(x) + x^2g(x) + x^3g(x) = x + x^5 + x^6 \Rightarrow v_{14} = (0\,1\,0\,0\,0\,1\,1)$$

$$g(x) + xg(x) + x^2g(x) + x^3g(x) = 1 + x^3 + x^5 + x^6$$

$$\Rightarrow v_{15} = (1\,0\,0\,1\,0\,1\,1)$$

The example above showed that g(x) generates the cyclic code $C(7,4)$ given in (Table 17). Hence we call it the generator polynomial of C.

Definition 5.2: The generator polynomial of a cyclic code C is the unique non zero polynomial of minimal degree r in C and is denoted by $g(x) = 1 + g_1x + \cdots + g_{r-1}x^{r-1} + x^r$.

Property 3: A binary polynomial of degree $\leq n-1$ is a code polynomial if and only if it is a multiple of $g(x)$.

Proof: ($\Leftarrow$) Let $v(x)$ be a binary polynomial of degree $n-1$ or less. Suppose that $v(x)$ is a multiple of $g(x)$. Then

$$v(x) = (a_0 + a_1 x + \cdots + a_{n-r-1}x^{n-r-1})g(x)$$
$$= a_0 g(x) + a_1 x g(x) + \cdots + a_{n-r-1}x^{n-r-1}g(x).$$

Since $v(x)$ is a linear combination of the code polynomials,

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

then $v(x)$ is a code polynomial in C.

($\Rightarrow$) Now, let $v(x)$ be a code polynomial in C.

Dividing $v(x)$ by $g(x)$, we obtain

$$v(x) = a(x).g(x) + b(x),$$

where either $b(x)$ is identical to zero or the degree of $b(x)$ is less than the degree of $g(x)$.

Rearranging the equation above, we have

$$b(x) = v(x) + a(x).g(x),$$

It follows from the first part that $a(x).g(x)$ is a code polynomial.

Since both $v(x)$ and $a(x).g(x)$ are code polynomials in linear code, then $b(x)$ must also be a code polynomial.

If $b(x) \neq 0$, then $b(x)$ is a nonzero code polynomial whose degree is less than the degree of $g(x)$.

This contradicts the assumption that $g(x)$ is the unique nonzero code polynomial of minimum degree. Thus, $b(x)$ must be identical to zero.

Hence, $v(x) = a(x).g(x)$. This proves that a code polynomial is a multiple of $g(x)$. ■

Lemma 5.1: There are 2^{n-r} distinct code polynomials of degree $\leq n-1$ in C.

Proof : Suppose $v(x)$ is a code polynomial of degree $\leq n-1$. Then by (Property 3) $v(x)$ is a multiple of $g(x) = 1 + g_1x + .. + g_{r-1}x^{r-1} + x^r$ and can be written as $v(x) = (a_0 + a_1x + \cdots + a_{n-r-1}x^{n-r-1})g(x) = a(x).g(x)$ for some polynomial $a(x)$.

Since $a_i = 0 \text{ or } 1$, then there are a total of 2^{n-r} distinct polynomials of degree $\leq n-r-1$.

Thus, the number of polynomials of degree $\leq n-1$ is also 2^{n-r}. ■

Note that, the 2^{n-r} polynomials given in lemma 5.1 form all code polynomials of the (n,k) cyclic code C.

Property 4: The degree of the generator polynomial of an (n,k) cyclic code is $n-k$.

Proof: Let $g(x) = 1 + g_1x + .. + x^r$ be the generator polynomial of a linear cyclic code $C(n,k)$, then from (Chapter 2) a linear $C(n,k)$ code has 2^k distinct codewords. from (Lemma 5.1) there are 2^{n-r} distinct code polynomials.

Now, by one to one corresponding of the codeword and code polynomial we have; $2^k = 2^{n-r} \Rightarrow n-r = k \Rightarrow r = n-k$ ■

At this point, a natural question is how to select a generator polynomial $g(x)$ which generates the (n, k) cyclic code C.

Theorem 5.3: The generator polynomial $g(x)$ of an (n, k) cyclic code is a factor of $x^n + 1$.

Proof: Multiplying $g(x)$ by x^k results in a polynomial $x^k.g(x)$ of degree n since $g(x)$ is a polynomial of degree $n-k$. Dividing $x^k.g(x)$ by $x^n + 1$, then we obtain the following equation from (5.6)

$$x^k g(x) = (g_{n-k} + g_{n-k+1}x + \cdots + g_{n-1}x^{k-1})(x^n + 1) + g^{(k)}(x)$$

where the code polynomial $g^{(k)}(x)$ is the remainder and it is obtained by shifting $g(x)$ to the right cyclically k times.

But $v_{n-k} = 1$ and $v_{n-k+1} = \cdots = v_{n-1} = 0$. Hence,

$$x^k g(x) = (x^n + 1) + g^{(k)}(x) \qquad (5.8)$$

Now, using (Property 3) $g^{(k)}(x)$ is a multiple of $g(x)$, say

$$g^{(k)}(x) = a(x).g(x)$$

From (5.8) we obtain

$$x^n + 1 = x^k g(x) + g^{(k)}(x) = x^k g(x) + a(x).g(x)$$
$$= \left(x^k + a(x)\right).g(x)$$

Thus, $g(x)$ is a factor of $x^n + 1$. ■

Definition 5.3: Let $f(x)$ be a fixed polynomial over $GF(2)$. Two polynomials $g(x)$ and $h(x)$ are said to be congruent modulo $f(x)$, written

$$g(x) \equiv h(x) \bmod f(x)$$

if $g(x) - h(x)$ is divisible by $f(x)$

Corollary 5.3: $x^n \equiv 1 \, mod \, g(x)$

Proof: By (Theorem 5.3) $x^n + 1 = g(x)q(x)$ for some $q(x)$ of degree k.

$\Rightarrow g(x)/x^n + 1 \Rightarrow x^n + 1 \equiv 0 \, mod \, g(x) \Rightarrow \; x^n \equiv 1 \, mod \, g(x)$. ■

Theorem 5.4: If $g(x)$ is a polynomial of degree $n-k$ that divides $x^n + 1$, then $g(x)$ generates an (n,k) cyclic code.

Proof: Let $g(x)$ be a polynomial of degree $n-k$ that divides $x^n + 1$ then $x^n + 1 = q(x).g(x)$, for some $q(x)$ of degree k. Consider the k polynomials $g(x), xg(x), \ldots, x^{k-1}g(x)$, which all have degree $\leq n-1$. A linear combination of these k polynomials is

$$v(x) = a_0 g(x) + a_1 x g(x) + \cdots + a_{k-1} x^{k-1} g(x)$$
$$= (a_0 + a_1 x + \cdots + a_{k-1} x^{k-1}) g(x)$$

is also a polynomial of degree $\leq n-1$ and is a multiple of $g(x)$.

There are a total of 2^k such polynomials and they form an (n,k) linear code C.

Now, we show this (n,k) linear code is cyclic.

Let $v(x) = v_0 + v_1 x + \cdots + v_{n-1} x^{n-1}$ be a code polynomial in this code. To complete this proof we must show the cyclic shift of $v(x)$ is also a code polynomial in C

Multiplying $v(x)$ by x, we obtain

$$xv(x) = v_0 x + v_1 x^2 + \cdots + v_{n-1} x^n$$
$$= (v_{n-1} + v_0 x + v_1 x^2 + \cdots + v_{n-2} x^{n-1}) + v_{n-1} x^n + v_{n-1}$$
$$= v^{(1)}(x) + v_{n-1}(x^n + 1) \qquad (5.9)$$

Rearranging the equation (5.9) we have

$$v^{(1)}(x) = xv(x) + v_{n-1}(x^n + 1)$$

$$= x(a_0 + a_1x + \cdots + a_{k-1}x^{k-1})g(x) + v_{n-1}.q(x).g(x)$$

$$= (a_0x + a_1x^2 + \cdots + a_{k-1}x^k)g(x) + v_{n-1}.q(x).g(x)$$

$$= g(x).[(a_0x + a_1x^2 + \cdots + a_{k-1}x^k) + v_{n-1}.q(x)] \qquad (5.10)$$

Equation (5.10) show that $v^{(1)}(x)$ is a multiple of $g(x)$.

But $q(x)$ is a polynomial of degree k, so it is can be written as follows;

$q(x) = u_0 + u_1x + \cdots + x^k$

Thus, (5.10) will be written as follows;

$$v^{(1)}(x) = g(x)[(a_0x + \cdots + a_{k-1}x^k) + v_{n-1}.(u_0 + u_1x + \cdots + x^k)]$$

$$= g(x)[(a_0x + \cdots + a_{k-1}x^k) + (v_{n-1}.u_0 + v_{n-1}u_1x + \cdots + v_{n-1}x^k)]$$

$$= g(x)v_{n-1}u_0 + g(x)\begin{bmatrix}(a_0 + v_{n-1}.u_1)x + (a_1 + v_{n-2}u_2)x^2 \\ + \cdots + (a_{k-1} + v_{n-1})x^k\end{bmatrix}$$

$$= g(x)v_{n-1}u_0 + (a_0 + v_{n-1}.u_1)xg(x) + (a_1 + v_{n-2}u_2)x^2g(x) + \cdots + (a_{k-1} + v_{n-1})x^kg(x)$$

(5.11)

The last equation (5.11) showed the cyclic shift $v^{(1)}(x)$ of $v(x)$ is a linear combination of $g(x), xg(x), \ldots, x^{k-1}g(x)$. Hence, $v^{(1)}(x)$ is also a code polynomial and the linear code generated by $g(x), xg(x), \ldots, x^{k-1}g(x)$ is an (n, k) cyclic code. ■

Note that, theorem 5.4 actually says that any factor $g(x)$ of $x^n + 1$ with degree $n - k$ generates an (n, k) cyclic code C, by taking all linear combination of $g(x), xg(x), \ldots, x^{k-1}g(x)$. In other words, the k code polynomials $g(x), xg(x), \ldots, x^{k-1}g(x)$ span C.

i.e., $C = \prec g(x), xg(x), \ldots, x^{k-1}g(x) \succ$

Example 5.5: The generator polynomial $g(x)$ of the (7, 4) cyclic code is a factor of $x^7 + 1$ and it is a polynomial of degree 3.

The polynomial $x^7 + 1$ can be factored as follows:

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

There are two factors of degree 3 each generates a (7, 4) cyclic code.

If $g(x) = 1 + x + x^3$, then from above we can design (7, 4) cyclic code C

where $C = < g(x), xg(x), \ldots, x^{k-1}g(x) >$ (5.12)

In fact, (5.12) generates the same cyclic code given in Example 5.1.

If $g(x) = 1 + x^2 + x^3$, then the cyclic code consists of

$v_0 = (0\ 0\ 0\ 0\ 0\ 0\ 0)$

$g(x) = 1 + x^2 + x^3 \Rightarrow v_1 = (1\ 0\ 1\ 1\ 0\ 0\ 0)$

$xg(x) = x + x^3 + x^4 \Rightarrow v_2 = (0\ 1\ 0\ 1\ 1\ 0\ 0)$

$x^2.g(x) = x^2 + x^4 + x^5 \Rightarrow v_3 = (0\ 0\ 1\ 0\ 1\ 1\ 0)$

$x^3.g(x) = x^3 + x^5 + x^6 \Rightarrow v_4 = (0\ 0\ 0\ 1\ 0\ 1\ 1)$

$g(x) + xg(x) = 1 + x + x^2 + x^4 \Rightarrow v_5 = (1\ 1\ 1\ 0\ 1\ 0\ 0)$

$g(x) + x^2g(x) = 1 + x^3 + x^4 + x^5 \Rightarrow v_6 = (1\ 0\ 0\ 1\ 1\ 1\ 0)$

$g(x) + x^3g(x) = 1 + x^2 + x^5 + x^6 \Rightarrow v_7 = (1\ 0\ 1\ 0\ 0\ 1\ 1)$

$xg(x) + x^2g(x) = x + x^2 + x^3 + x^5 \Rightarrow v_8 = (0\ 1\ 1\ 1\ 0\ 1\ 0)$

$xg(x) + x^3g(x) = x + x^4 + x^5 + x^6 \Rightarrow v_9 = (0\ 1\ 0\ 0\ 1\ 1\ 1)$

$x^2g(x) + x^3g(x) = x^2 + x^3 + x^4 + x^6 \Rightarrow v_{10} = (0\ 0\ 1\ 1\ 1\ 0\ 1)$

$g(x) + xg(x) + x^2g(x) = 1 + x + x^5 \Rightarrow v_{11} = (1\ 1\ 0\ 0\ 0\ 1\ 0)$

$g(x) + xg(x) + x^3g(x) = 1 + x + x^2 + x^3 + x^4 + x^5 + x^6$

$\Rightarrow v_{12} = (1\ 1\ 1\ 1\ 1\ 1\ 1)$

$g(x) + x^2g(x) + x^3g(x) = 1 + x^4 + x^6 \Rightarrow v_{13} = (1\ 0\ 0\ 0\ 1\ 0\ 1)$

$$xg(x) + x^2 g(x) + x^3 g(x) = x + x^2 + x^6 \Longrightarrow v_{14} = (0\ 1\ 1\ 0\ 0\ 0\ 1)$$

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

$$\Longrightarrow v_{15} = (1\ 1\ 0\ 1\ 0\ 0\ 1)$$

Note that, in the above example we can find a (7, 3) cyclic code generated by the generator polynomial $g(x) = (1 + x)(1 + x + x^3) = 1 + x^2 + x^3 + x^4$ in the same way.

5.4 The Generating Matrix, the Check Polynomial and the Parity Check Matrix for Cyclic Codes

5.4.1 The Generator Matrix

Recall that the k code polynomials $g(x), xg(x), \ldots, x^{k-1}g(x)$ span an (n, k) cyclic code C with generator polynomial $g(x) = g_0 + g_1 x + \cdots + g_{n-k}x^{n-k}$. Then, if the k n-tuples corresponding to these k code polynomials are used as the rows of an kxn matrix, we obtain the following generator matrix for C:

$$G = \begin{bmatrix} g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} & 0 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} & 0 & \cdot & \cdot & 0 \\ \mathrm{M} & & & & & & & & & & & & & & \mathrm{M} \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & g_0 & g_1 & g_2 & \cdot & \cdot & \cdot & \cdot & \cdot & g_{n-k} \end{bmatrix}$$

Note that, all rows of G are linearly independent. So, $g(x), xg(x), \ldots, x^{k-1}g(x)$ form the basis for C.

Example 5.6: Design all cyclic codes of length 4 using the generator polynomial and generator matrix for each cyclic code.

Factorization of $x^4 + 1$ over $GF(2)$ has the form

$$x^4 + 1 = (x + 1)(x^3 + x^2 + x + 1) = (x + 1)(x + 1)(x^2 + 1)$$

Table 18: All Cyclic Codes of Length 4

Generator polynomial $g(x)$	Generator matrix $G(x)$
1	I_4
$1 + x$	$\begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{bmatrix}$
$1 + x^2$	$\begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix}$
$x^3 + x^2 + x + 1$	$[1\ 1\ 1\ 1]$
$x^4 + 1$	$[0\ 0\ 0\ 0]$

Note that, $x^4 + 1 \equiv 0 \ mod \ x^4 + 1$.

5.4.2 Check Polynomials

Let C be a cyclic (n, k) code with the generator polynomial $g(x)$ (of degree $n - k$), then $x^n + 1 = g(x)h(x)$ (5.13)

where the polynomial $h(x)$ has the degree k and is of the following form:

$h(x) = h_0 + h_1 x + \cdots + h_k x^k$

with $h_0 = h_k = 1$.

We call $h(x)$ the check polynomial of C or the parity polynomial of C.

The check polynomial will be used to determine if the received word is a codeword in the cyclic code C as follows;

Theorem 5.5: Let $C(n,k)$ be a cyclic code with a generator polynomial $g(x)$ and check polynomial $h(x)$. Then $v(x) \in C$ if and only if $v(x)h(x) \equiv 0 \; mod \; x^n + 1$.

Proof: ($\Longrightarrow$) Suppose $v(x) \in C \Rightarrow v(x) = a(x)g(x)$ (5.14)

for some $a(x) \in V_n$.

Multiply both sides of (5.14) by $h(x)$ then we obtain

$v(x)h(x) = a(x)g(x)h(x) = a(x)(x^n + 1) \equiv 0 \; mod \; (x^n + 1)$

($\Longleftarrow$) Suppose $v(x)h(x) \equiv 0 \; mod \; x^n + 1$. To show that $v(x) \in C$, we must show $v(x)$ is a multiple of $g(x)$ by (Property 3 of $g(x)$).

Consider $v(x) = q(x)g(x) + r(x), \deg r(x) < \deg g(x)$.

where $q(x) \& r(x)$ are the quotient and remainder polynomials, respectively.

Multiply both sides of the above equation by $h(x)$ then we obtain

$v(x)h(x) = q(x)g(x)h(x) + r(x)h(x)$

But $v(x)h(x) \equiv 0 \; mod \; x^n + 1 \; \& \; g(x)h(x) = x^n + 1 \equiv 0 \; mod \; x^n + 1$

by (5.13)

$\Rightarrow r(x)h(x) \equiv 0 \; mod \; x^n + 1 \Rightarrow r(x) = 0$ or $h(x) = 0$ but $h(x) \neq 0$, so $r(x) = 0$. Therefore $v(x) = q(x)g(x)$ is a multiple of $g(x)$. ■

Remark 5.4: The check polynomial of the cyclic code $C(n,k)$ generated by $g(x)$ is $h(x) = \frac{(x^n+1)}{g(x)}$.

Proof: From (5.13).

Example 5.7: Consider the (7, 4) cyclic code generated by $g(x) = 1 + x + x^3$ with the parity check polynomial $h(x) = 1 + x + x^2 + x^4$.

Then the polynomial $r(x) = x + x^5 + x^6$ is a code polynomial of C if and only if $r(x)h(x) \equiv 0 \ mod \ x^7 + 1$.

Since $r(x).h(x) = (x + x^5 + x^6)(1 + x + x^2 + x^4)$

$$= x + x^2 + x^3 + x^8 + x^9 + x^{10}$$

and

$$\begin{array}{r|l} x^7 + 1 & x^3 + x^2 + x \\ \hline x^{10} + x^9 + x^8 + x^3 + x^2 + x & \\ x^{10} + x^3 & \\ \hline x^9 + x^8 + x^2 + x & \\ x^9 + x^2 & \\ \hline x^8 + x & \\ x^8 + x & \\ \hline 0 + 0 & \end{array}$$

Thus $r(x)h(x) \equiv 0 mod \ x^7 + 1 \Rightarrow r(x)$ is a code polynomial of $C(7,4)$.

5.4.3 Parity Check Matrices

In this section we will use the check polynomial $h(x)$ of the cyclic code C to generates the dual code $C^{\perp}$.

Theorem 5.6: Suppose C is a cyclic (n, k) code with the check polynomial $h(x) = 1 + h_1 x + \cdots + x^k$, then

(i) The parity check matrix for C is

$$H = \begin{bmatrix} h_k & h_{k-1} & \mathbf{L} & h_0 & 0 & \mathbf{L} & 0 \\ 0 & h_k & \mathbf{L} & h_1 & h_0 & \mathbf{L} & 0 \\ \mathbb{M} & \mathbf{L} & & & & & \\ 0 & 0 & \mathbf{L} & 0 & h_k & \mathbf{L} & h_0 \end{bmatrix}$$

(ii) The polynomial

$$\overline{h}(x) = x^k h(x^{-1}) = h_k + h_{k-1}x + \cdots + h_0 x^k \qquad (5.15)$$

generates an $(n, n-k)$ cyclic code $C^{\perp}$.

Note: the polynomial $\overline{h}(x)$ is defined as the reciprocal of $h(x)$

Proof:

(i) We show that any codeword $v = (v_0, v_1, \ldots, v_{n-1})$ in C is orthogonal to every row of H.

Let $v(x) = v_0 + v_1 x + \cdots + v_{n-1}x^{n-1}$ be a code polynomial in C. Then $v(x) = a(x).g(x)$ for some polynomial $a(x)$ of degree $\leq k-1$.

Multiplying $v(x)$ by $h(x)$, we obtain

$$v(x)h(x) = a(x)g(x)h(x)$$

$$= a(x)(x^n + 1)$$

$$= a(x) + x^n a(x) \qquad (5.16)$$

$$= (a_0 + a_1 x + \cdots + a_{k-1}x^{k-1}) + a_0 x^n + a_1 x^{n+1} + \cdots + a_{k-1}x^{n+k-1}$$

Clearly the powers $x^k, x^{k+1}, \ldots, x^{n-1}$ do not appear in (5.16). If we expand the product $v(x)h(x)$ on the left-hand side of (5.16), the coefficients of $x^k, x^{k+1}, \ldots, x^{n-1}$ must be equal to zero, i.e.

$$\begin{aligned} v_0 h_k + v_1 h_{k-1} + \cdots + v_k h_0 &= 0 \\ v_1 h_k + v_2 h_{k-1} + \cdots + v_{k+1} h_0 &= 0 \\ &\vdots \\ v_{n-k-1} h_k + v_{n-k} h_{k-1} + \cdots + v_{n-1} h_0 &= 0 \end{aligned} \qquad (5.17)$$

The system (5.17) can be represent as follows

$$\begin{bmatrix} h_k & h_{k-1} & \mathbf{L} & h_0 & 0 & \mathbf{L} & 0 \\ 0 & h_k & \mathbf{L} & h_1 & h_0 & \mathbf{L} & 0 \\ \mathbf{M} & \mathbf{L} & & & & & \\ 0 & 0 & \mathbf{L} & 0 & h_k & \mathbf{L} & h_0 \end{bmatrix} \begin{bmatrix} v_0 \\ v_1 \\ \mathbf{M} \\ v_{n-1} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \mathbf{M} \\ 0 \end{bmatrix}$$

It follows from (5.17) that any codeword $v = (v_0\ v_1\ \ldots v_{n-1}) \in C$ is orthogonal to the word $h = (h_k\ h_{k-1} \ldots h_0\ 0 \ldots 0)$ and to the any cyclic shift of h, i.e. any codeword $v \in C$ is orthogonal to every row of H. Therefore, H is a parity-check matrix of the cyclic code C. ■

Proof (ii): To show that $C^{\perp}$ is an $(n, n-k)$ cyclic code generated by the polynomial $\overline{h}(x) = h_k + h_{k-1}x + \cdots + h_0x^k$, it is sufficient to show that $\overline{h}(x)$ is a factor of $x^n + 1$.

Observe that from (5.13)

$$g(x)h(x) = x^n + 1 \tag{5.18}$$

Now, if we substituted x^{-1} in (5.18), then we obtain

$$g(x^{-1})h(x^{-1}) = (x^{-1})^n + 1 = x^{-n} + 1 \tag{5.19}$$

Multiplying both sides of (5.19) by x^n

$$x^n g(x^{-1})h(x^{-1}) = (x^{-1})^n + 1 = (x^{-n} + 1)x^n = 1 + x^n \tag{5.20}$$

Rewrite (5.20) using $x^n = x^{n-k+k} = x^{n-k}x^k$ in the left hand side of (5.20)

$$x^k h(x^{-1})x^{n-k} g(x^{-1}) = x^n + 1$$

But $\overline{h}(x) = x^k h(x^{-1})$ so we have the following

$$\overline{h}(x)[x^{n-k} g(x^{-1})] = x^n + 1$$

Therefore $\overline{h}(x)$ is indeed a factor of $x^n + 1$ and hence, the polynomial $\overline{h}(x)$ generates an $(n, n-k)$ cyclic code $C^{\perp}$. ■

Note that the row space of H is the dual $C^{\perp}$ of C.

Moreover, since the parity check matrix H is obtained from the polynomial $h(x)$, we call $h(x)$ the parity polynomial *of* C. Hence, a cyclic code is also specified by its parity polynomial.

Definition 5.4: In an (n, k) cyclic code C the minimum distance (The minimum weight) of the dual code $C^{\perp}$ is the degree of the polynomial $\overline{h}(x) = x^k h(x^{-1})$.

Example 5.8: Consider the (7,4) cyclic code given in table 17 with generator polynomial $g(x) = 1 + x + x^3$ and generator matrix

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

Then the parity polynomial $h(x)$ is

$$h(x) = \frac{(x^n + 1)}{g(x)} = \frac{x^7 + 1}{1 + x + x^3} = 1 + x + x^2 + x^4$$

and hence, the reciprocal of $h(x)$ is

$$\overline{h}(x) = x^k h(x^{-1}) = x^4 h(x^{-1}) = x^4(1 + x^{-1} + x^{-2} + x^{-4})$$

$$= 1 + x^2 + x^3 + x^4$$

The parity check matrix H of $C(7,4)$ is

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

Now, The dual code $C^{\perp}(7,3)$ of C, which is generated by $\overline{h}(x) = 1 + x^2 + x^3 + x^4$ consists of

$v_0 = (0\,0\,0\,0\,0\,0\,0)$

$\overline{h}(x) = 1 + x^2 + x^3 + x^4 \Rightarrow v_1 = (1\,0\,1\,1\,1\,0\,0)$

$x\overline{h}(x) = x + x^3 + x^4 + x^5 \Rightarrow v_2 = (0\,1\,0\,1\,1\,1\,0)$

$x^2\overline{h}(x) = x^2 + x^4 + x^5 + x^6 \Rightarrow v_3 = (0\,0\,1\,0\,1\,1\,1)$

$\overline{h}(x) + x\overline{h}(x) = 1 + x + x^2 + x^5 \Rightarrow v_4 = (1\,1\,1\,0\,0\,1\,0)$

$\overline{h}(x) + x^2\overline{h}(x) = 1 + x^3 + x^5 + x^6 \Rightarrow v_5 = (1\,0\,0\,1\,0\,1\,1)$

$x\overline{h}(x) + x^2\overline{h}(x) = x + x^2 + x^3 + x^6 \Rightarrow v_6 = (0\,1\,1\,1\,0\,0\,1)$

$\overline{h}(x) + x\overline{h}(x) + x^2\overline{h}(x) = 1 + x + x^4 + x^6 \Rightarrow v_7 = (1\,1\,0\,0\,1\,0\,1)$

Clearly, the minimum distance (minimum weight) of $C^{\perp}$ is 4, which is the degree of the polynomial $\overline{h}(x) = 1 + x^2 + x^3 + x^4$.

The parity check matrix H can be used to check the codewords of the cyclic code C as in the following theorem:

Theorem 5.8: r is a codeword in a cyclic code $C(n,k)$ if and only if $H_{(n-k)\times n} r^T = 0$.

This can be expressed as

$$\begin{bmatrix} h_k & h_{k-1} & \mathbf{L} & h_0 & 0 & \mathbf{L} & 0 \\ 0 & h_k & \mathbf{L} & h_1 & h_0 & \mathbf{L} & 0 \\ \mathbf{M} & \mathbf{L} & & & & & \\ 0 & 0 & \mathbf{L} & 0 & h_k & \mathbf{L} & h_0 \end{bmatrix} \begin{bmatrix} r_0 \\ r_1 \\ \mathbf{M} \\ r_{n-1} \end{bmatrix} = 0$$

Proof: It follows from theorem 5.6 (i) that any codeword $v = (v_0, v_1, \ldots, v_{n-1})$ in $C(n,k)$ is orthogonal to every row of H. ■

5.4.4 Systematic Form of G & H

The generator matrix in systematic form can also be formed easily using these three steps:

Consider $\mathcal{C}(n,k)$ is the cyclic code with the generator polynomial $g(x)$.

First: Dividing x^{n-k+i} by the generator polynomial $g(x)$ for $i = 0,1,\dots,k-1$, we obtain

$$x^{n-k+i} = a_i(x)g(x) + b_i(x), \tag{5.21}$$

where $b_i(x)$ is the remainder with the following form:

$$b_i(x) = b_{i0} + b_{i1}x + \cdots + b_{i,n-k-1}x^{n-k-1}$$

Second: Rearranging (5.21), for $i = 0,1,\dots,k-1$, we obtain the following

$$b_i(x) + x^{n-k+i} = a_i(x)g(x) \tag{5.22}$$

Since the R.H.S. of (5.22) is a multiple of $g(x)$ for $i = 0,\dots,k-1$, then the L.H.S. $b_i(x) + x^{n-k+i}$ is a code polynomial v_i in $\mathcal{C}$.

Third: Arranging the k coefficients of the left hand side of (5.22) as rows of a $k \times n$ matrix as follows

$$G^* = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \mathbf{L} & b_{0,n-k-1} & 1 & 0 & 0 & \mathbf{L} & 0 \\ b_{10} & b_{11} & b_{12} & \mathbf{L} & b_{1,n-k-1} & 0 & 1 & 0 & \mathbf{L} & 0 \\ b_{20} & b_{21} & b_{22} & \mathbf{L} & b_{2,n-k-1} & 0 & 0 & 1 & & 0 \\ \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & \mathrm{M} & & \mathrm{M} & \\ b_{k-1,0} & b_{k-1,1} & b_{k-1,2} & \mathbf{L} & b_{k-1,n-k-1} & 0 & 0 & 0 & \mathbf{L} & 1 \end{bmatrix} = \left(B_{k\times(n-k)} \middle| I_k\right)$$

which is the generator matrix of $\mathcal{C}$ in the systematic form. Since the rows of G^* span the row space of it, where any linear combination of these rows is again codeword in the row space.

$\Longrightarrow \sum_{i=1}^{k} c_i R_i$ is also a codeword, where R_i is i^{th} row in G^*.

$c_i = 0 \ \ or \ \ 1 \Longrightarrow 2^k$ codewords.

But $|C| = 2^k \Longrightarrow C$ is spanned by the row space of G^*.

Moreover, the rows of G^* are linearly independent.

$\Longrightarrow G^*$ generates C.

The corresponding parity check matrix for C is the following $(n-k)xn$ matrix; $H^* = \left(I_{n-k} \middle| B^T{}_{(n-k)\times k}\right)$

$$= \begin{bmatrix} 1 & 0 & 0 & \mathbf{L} & 0 & b_{00} & b_{10} & b_{20} & \mathbf{L} & b_{k-1,0} \\ 0 & 1 & 0 & \mathbf{L} & 0 & b_{01} & b_{11} & b_{21} & \mathbf{L} & b_{k-1,1} \\ 0 & 0 & 1 & & 0 & b_{02} & b_{12} & b_{22} & \mathbf{L} & b_{k-1,2} \\ \mathbf{M} & \mathbf{M} & & & \mathbf{M} & \mathbf{M} & \mathbf{M} & \mathbf{M} & & \mathbf{M} \\ 0 & 0 & 0 & \mathbf{L} & 1 & b_{0,n-k-1} & b_{1,n-k-1} & b_{2,n-k-1} & \mathbf{L} & b_{k-1,n-k-1} \end{bmatrix}$$

Example 5.9: Again, let us consider the (7, 4) cyclic code given in table 17 which is generated by $g(x) = 1 + x + x^3$.

Clearly, the generator matrix G and the parity check matrix H of C given in example 5.8 are not in systematic form so;

Dividing $x^3, x^4, x^5,$ and x^6 by $g(x)$, we have

$x^3 = g(x) + (1 + x),$

$x^4 = xg(x) + (x + x^2),$

$x^5 = (x^2 + 1)g(x) + (1 + x + x^2),$

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

Rearranging the equations above using (5.24), we obtain the following four code polynomials:

$$v_0(x) = 1 + x + x^3,$$

$$v_1(x) = x + x^2 + x^4,$$

$$v_2(x) = 1 + x + x^2 + x^5,$$

$$v_3(x) = 1 + x + x^2 + x^5,$$

$$v_3(x) = 1 + x^2 + x^6.$$

Arranging the coefficients of these four code polynomials as rows of the 4×7 matrix

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

which is the generator matrix of $C(7,4)$ in systematic form.

The corresponding parity check matrix for $C(7,4)$ is

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

Important fact: The above algorithm can be made easier when it uses the power representation for the elements of C given in section 1.8.

Given $g(x) = 1 + g_1 x + \cdots + x^{n-k}$ as a primitive polynomial which has a zero in $GF(2^{n-k})$ at the primitive element α, and thus all codewords v satisfy $v(\alpha) = a(\alpha)g(\alpha) = a(\alpha).0 = 0$.

That is $1 + g_1\alpha + \cdots + \alpha^{n-k} = 0$

$$\overset{then}{\Longrightarrow} \alpha^{n-k} = 1 + g_1\alpha + \cdots + g_{n-k-1}\alpha^{n-k-1} \qquad (5.23)$$

And the parity check matrix $H = [\alpha^0 \ \alpha^1 \ \ldots \ \alpha^{n-1}]$

For example: The above generator matrix G^* given in example 5.9 can be formed in another way using (5.23).

Since the polynomial $g(x) = 1 + x + x^3$ is a primitive polynomial which has a zero in $GF(2^3)$ at α

$$1 + \alpha + \alpha^3 = 0 \overset{then}{\Longrightarrow} \alpha^3 = 1 + \alpha \Rightarrow b_0(x) = 1 + x$$

$\Rightarrow v_0(x) = 1 + x + x^3 \Rightarrow v_0 = (1\ 1\ 0\ 1\ 0\ 0\ 0)$

$\alpha^4 = \alpha(\alpha^3) = \alpha(1+\alpha) = \alpha + \alpha^2 \Rightarrow b_1(x) = x + x^2$

$\Rightarrow v_1(x) = x + x^2 + x^4 \Rightarrow v_1 = (0\ 1\ 1\ 0\ 1\ 0\ 0)$

$\alpha^5 = \alpha(\alpha^4) = \alpha(\alpha + \alpha^2) = 1 + \alpha + \alpha^2 \Rightarrow b_2(x) = 1 + x + x^2$

$\Rightarrow v_2(x) = 1 + x + x^2 + x^5 \Rightarrow v_2 = (1\ 1\ 1\ 0\ 0\ 1\ 0)$

$\alpha^6 = \alpha(\alpha^5) = 1 + \alpha^2 \Rightarrow b_3(x) = 1 + x^2$

$\Rightarrow v_3(x) = 1 + x^2 + x^6 \Rightarrow v_3 = (1\ 0\ 1\ 0\ 0\ 0\ 1)$

where $b_0, b_1, b_2 \& b_3$ are the remainders when we divide x^3, x^4, x^5 and x^6 by $g(x)$, respectively.

Arranging the coefficients of these four polynomials $v_0(x), v_1(x), v_2(x) \& v_3(x)$ as rows of the $4x7$ matrix

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

which is the generator matrix of $C(7,4)$ in systematic form.

$$\& H = [\alpha^0\ \ \alpha^1\ \ \alpha^2\ \ \alpha^3\ \ \alpha^4\ \ \alpha^5\ \ \alpha^6] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix}.$$

5.5 Encoding Operations

5.5.1 Nonsystematic Encoding

In an (n, k) cyclic code, every code polynomial $v(x)$ can be expressed as a multiple the message $u(x)$ and $g(x) = 1 + g_1 x + \cdots + g_{r-1} x^{r-1} + x^r$ where the degree of $g(x)$ is $r = n - k$.

$$v(x) = u(x).g(x) \qquad (5.24)$$

$$= (u_0 + u_1 x + \cdots + u_{k-1} x^{k-1}).g(x)$$

$$= (u_0 g(x) + u_1 x g(x) + \cdots + u_{k-1} x^{k-1} g(x)).$$

$$= (u_0\ u_1\ \ldots u_{k-1}) \begin{bmatrix} g(x) \\ x g(x) \\ \vdots \\ x^{k-1} g(x) \end{bmatrix}$$

Using the vectors the code polynomial $v(x)$ given in (5.24) can be expressed as follows;

$$v = (u_0\ u_1 \ldots u_{k-1}) \begin{bmatrix} g_0 & g_1 & \cdots & g_r & & & & \\ & g_0 & g_1 & \cdots & g_r & & & \\ & & g_0 & g_1 & \cdots & g_r & & \\ & & & \ddots & \ddots & & \ddots & \\ & & & & g_0 & g_1 & \cdots & g_r \\ & & & & & g_0 & g_1 & \cdots & g_r \end{bmatrix} \quad (5.25)$$

$v = u.\, G_{kxn}$

where u and v correspond to the $u(x)$ and $v(x)$, respectively.

Therefore, an (n, k) cyclic code is completely specified by its generator polynomial $g(x)$.

Remark 5.5: The degree of $g(x)$ is equal to the number of parity-check digits $n - k$ of the code.

Example 5.10: Consider the cyclic code $\mathcal{C}(7,4)$ generated by

$g(x) = 1 + x + x^3$, where message set is

$U = \{(0000), (1000), (0100), (1100), (0010), (1010), (0110), (1110),$

$(0001), (1001), (0101), (1101), (0011), (1011), (0111), (1111)\}$.

Using (5.24), the corresponding encoded messages are given in the following table.

Table 19: The Cyclic Code $C(7,4)$ Generated by $g(x) = 1 + x + x^3$

Messages u	Message polynomials $u(x)$	Code polynomials $v(x) = u(x).g(x)$ (By 5.24)	Codewords v (By 5.25)
(0000)	0	0=0.$g(x)$	(0000000)
(1000)	1	$1 + x + x^3$	(1101000)
(0100)	x	$x + x^2 + x^4$	(0110100)
(1100)	$1 + x$	$1 + x^2 + x^3 + x^4$	(1011100)
(0010)	x^2	$x^2 + x^3 + x^5$	(0011010)
(1010)	$1 + x^2$	$1 + x + x^2 + x^5$	(1110010)
(0110)	$x + x^2$	$x + x^3 + x^4 + x^5$	(0101110)
(1110)	$1 + x + x^2$	$1 + x^4 + x^5$	(1000110)
(0001)	x^3	$x^3 + x^4 + x^6$	(0001101)
(1001)	$1 + x^3$	$1 + x + x^4 + x^6$	(1100101)
(0101)	$x + x^3$	$x + x^2 + x^3 + x^6$	(0111001)
(1101)	$1 + x + x^3$	$1 + x^2 + x^6$	(1010001)
(0011)	$x^2 + x^3$	$x^2 + x^4 + x^5 + x^6$	(0010111)
(1011)	$1 + x^2 + x^3$	$1 + x + x^2 + x^3 + x^4 + x^5 + x^6$	(1111111)
(0111)	$x + x^2 + x^3$	$x + x^5 + x^6$	(0100011)
(1111)	$1 + x + x^2 + x^3$	$1 + x^3 + x^5 + x^6$	(1001011)

5.5.2 Systematic Encoding

In systematic encoding the codeword v is divided into two parts:

the message part and the redundant checking part. The message part consists of the k message digits u and the redundant checking part consists of $n - k$ parity-check digits.

Encoding in systematic form consists of three steps:

Consider the message vector u and the corresponding message polynomial $u(x)$.

$$u = (u_0\ u_1\ \dots\ u_{k-1}) \leftrightarrow u(x) = u_0 + u_1 x + \dots + u_{k-1}x^{k-1}.$$

Step 1: Shift u to the right $n-k$ positions by multiplying $u(x)$ by x^{n-k} to obtain a polynomial of degree $\leq n-1$.

$$x^{n-k}u(x) = u_0x^{n-k} + u_1x^{n-k+1} + \cdots + u_{k-1}x^{n-1} \qquad (5.26)$$

Observe that the vector corresponding to (5.26) is

$$(\underbrace{0\ 0\ \ldots\ 0}_{n-k}\ u_0\ u_1\ u_{k-1})$$

Step 2: Dividing $x^{n-k}u(x)$ by the generator polynomial $g(x)$ to obtain the remainder $b(x)$ (the parity-check digits).

$$x^{n-k}u(x) = a(x)g(x) + b(x) \qquad (5.27)$$

Where $a(x)$ and $b(x)$ are the quotient and the remainder, respectively.

Since the degree of $g(x)$ is $n-k$, the degree of $b(x)$ must be $n-k-1$ or less, that is,

$$b(x) = b_0 + b_1x + \cdots + b_{n-k-1}x^{n-k-1} \leftrightarrow (b_0\ b_1\ b_{n-k-1}\ \underbrace{0\ 0\ \ldots\ 0}_{k})$$

Step 3: Obtain the code polynomial $v(x)$ by adding $b(x)$ to $x^{n-k}u(x)$ as follows

$$\begin{aligned} v(x) = b(x) + x^{n-k}u(x) &= a(x)g(x) \\ &= b_0 + b_1x + \cdots + b_{n-k-1}x^{n-k-1} \\ &\quad + u_0x^{n-k} + u_1x^{n-k+1} + \cdots + u_{k-1}x^{n-1} \end{aligned}$$

which corresponds to the codeword

$$(b_0\ b_1 \ldots b_{n-k-1}\ u_0\ u_1\ \ldots u_{k-1}).$$

Recall that the division by the generator polynomial $g(x)$ of the above algorithm becomes easier when it is use the power representation of the elements of $\mathcal{C}$.

For $g(x) = 1 + g_1x + \cdots + x^{n-k}$ if $1 + g_1x + \cdots + x^{n-k} = 0$

$$\overset{then}{\Longrightarrow} x^{n-k} = 1 + g_1 x + \cdots + g_{n-k-1} x^{n-k-1} \qquad (5.28)$$

Example 5.12: The systematic form of the 16 codewords in the (7, 4) cyclic code which is given in (Table 19) are listed in the following table.

For the message polynomial $u(x) = u_0 + u_1 x + u_2 x^2 + u_3 x^3$ there is $b(x) = b_0 + b_1 x + b_2 x^2$ s.t. $b(x) + x^3 u(x)$ is a codeword in $C(7,4)$.

Table 20: The (7,4) Cyclic Code Generated by $g(x) = 1 + x + x^3$ in Systematic form

Message u	Message polynomial $u(x)$	Codeword v	Code polynomial $v(x)$
(0000)	0	(0000000)	0
(1000)	1	(1101000)	$1 + x + x^3$
(0100)	x	(0110100)	$x + x^2 + x^4$
(1100)	$1 + x$	(1011100)	$1 + x^2 + x^3 + x^4$
(0010)	x^2	(1110010)	$1 + x + x^2 + x^5$
(1010)	$1 + x^2$	(0011010)	$x^2 + x^3 + x^5$
(0110)	$x + x^2$	(1000110)	$1 + x^4 + x^5$
(1110)	$1 + x + x^2$	(0101110)	$x + x^3 + x^4 + x^5$
(0001)	x^3	(1010001)	$1 + x^2 + x^6$
(1001)	$1 + x^3$	(0111001)	$x + x^2 + x^3 + x^6$
(0101)	$x + x^3$	(1100101)	$1 + x + x^4 + x^6$
(1101)	$1 + x + x^3$	(0001101)	$x^3 + x^4 + x^6$
(0011)	$x^2 + x^3$	(0100011)	$x + x^5 + x^6$
(1011)	$1 + x^2 + x^3$	(1001011)	$1 + x^3 + x^5 + x^6$
(0111)	$x + x^2 + x^3$	(0010111)	$x^2 + x^4 + x^5 + x^6$
(1111)	$1 + x + x^2 + x^3$	(1111111)	$1 + x + x^2 + x^3 + x^4 + x^5 + x^6$

For instance, let $u(x) = 1 + x^2 + x^3 \leftrightarrow u = (1\ 0\ 1\ 1)$ be the message to be encoded.

Step 1; Multiplying $u(x)$ by x^3;

$$u(x).x^3 = x^3 + x^5 + x^6 \leftrightarrow (0\ 0\ 0\ 1\ 0\ 1\ 1)$$

Step 2; Dividing $u(x).x^3$ by the generator polynomial $g(x) = 1 + x + x^3$.

$$
\begin{array}{r}
x^3 + x^2 + x + 1 \\
x^3 + x + 1 \overline{)\, x^6 + x^5 + x^3} \\
\underline{x^6 + x^4 + x^3} \\
x^5 + x^4 \\
\underline{x^5 + x^3 + x^2} \\
x^4 + x^3 + x^2 \\
\underline{x^4 + x^2 + x} \\
x^3 + x \\
\underline{x^3 + x + 1} \\
0 + 0 + 1
\end{array}
$$

We obtain the remainder $b(x) = 1$. Thus, the code polynomial is

$v(x) =$

$$b(x) + x^3u(x) = 1 + x^3 + x^5 + x^6 \leftrightarrow v = (1\ 0\ 0\ \underbrace{1\ 0\ 1}_{u}\ 1).$$

Note that, if $x^3 + x + 1 = 0$ then $x^3 = x + 1$

For instance, if $u(x) = 1$ then $x^3u(x) = x^3 = 1 + x \Longrightarrow b(x) = 1 + x$

$\Longrightarrow v(x) = b(x) + x^3u(x) = 1 + x + x^3 \Rightarrow v = (1\ 1\ 0\ 1\ 0\ 0\ 0).$

if $u(x) = 1 + x^2 + x^3$ then $x^3u(x) = x^3 + x^5 + x^6 = (1 + x) + x^2(1 + x) + (1 + x)^2 = 1 + x + x^2 + x + 1 + 1 + x^2 = 1 \Longrightarrow b(x) = 1$

$\Longrightarrow v(x) = b(x) + x^3u(x) = 1 + x^3 + x^5 + x^6 \Rightarrow v = (1\ 0\ 0\ 1\ 0\ 1\ 1).$

Etc.

5.6 Shift-Register Encoders for Cyclic Codes

In this section we present circuits for performing the encoding operation by presenting circuits for computing polynomial multiplication and division. Hence, we shall show that every cyclic code can be encoded with a simple finite-state machine called a shift-register encoder.

To define the shift register we want to the following definition;

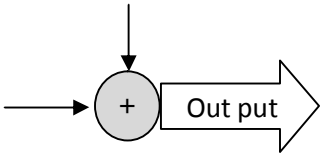
Definition 5.5: A D flip-flop is a one-bit memory storage in the field $GF(2)$.

Figure 15: Flip-Flop

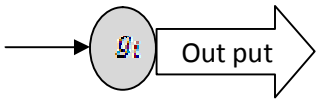
External clock: Not pictured in our simplified circuit diagrams, but an important part of them, which generates a timing signal ("tick") every t_0 seconds.

When the clock ticks, the content of each flip-flop is shifted out of the flip-flop in the direction of the arrow, through the circuit to the next flip-flop. The signal then stops until the next tick.

Adder: The symbol of adder has two inputs and one output, which is computed as the sum of the inputs (modulo 2-addition).



Multiplication: The symbol of multiplication has one input and one output, where the output is the multiplication of the input and the number g_i which is stored in this symbol (either 1 or 0), where 0 represented by no connection and 1 by a connection.



Definition 5.6: A shift-register is a chain of $(n-k)$ D flip-flops connected to each other, where the output from one flip-flop becomes the input of the next flip-flop.

Figure 16: Shift Register

All the flip-flops are driven by a common clock, and all are set or reset simultaneously.

5.6.1 Nonsystematic Encoder

Recall that for an (n,k) cyclic code C the code polynomial $v(x)$ corresponding to the message $u(x)$ is obtained by the encoding operation of polynomial multiplication:

$$v(x) = u(x)g(x).$$

It turns out that polynomial multiplication is easy to implement using the shift register encoder, and we shall now make a brief study of this subject using the following example.

Example 5.13: Consider a generator polynomial for a (7, 4) binary cyclic code of is $g(x) = g_0 + g_1x + g_2x^2 + g_3x^3$.

Then the corresponding shift-register encoder of the polynomial multiplication

$$v(x) = u(x).g(x)$$

$$= (u_0 + u_1x + u_2x^2 + u_3x^3)(g_0 + g_1x + g_2x^2 + g_3x^3) \qquad (5.29)$$

is shown in figure 17.

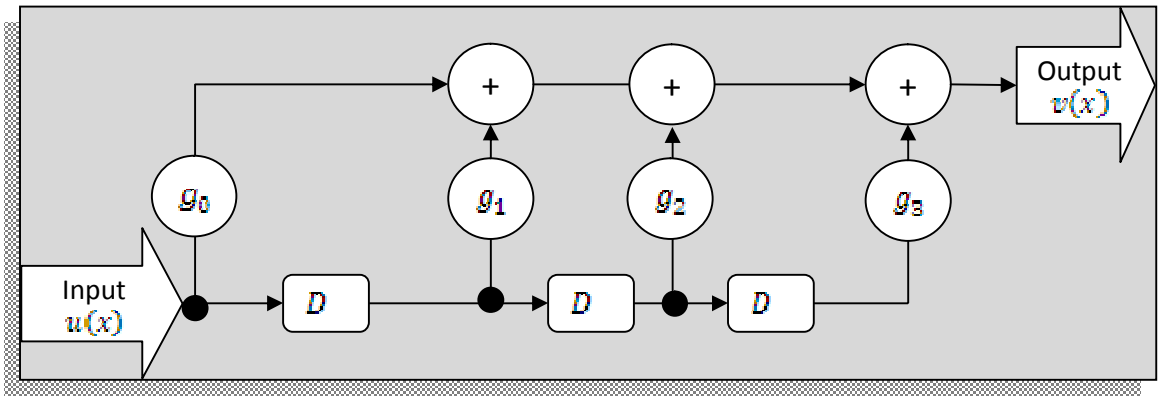


Figure 17: Nonsystematic Encoder for (7, 4) Cyclic Code with Generator Polynomial $g(x) = g_0 + g_1x + g_2x^2 + g_3x^3$.

Hear we have 3 flip-flops since $n = 7$ and $k = 4$.

Now let's understand why the circuit of (Figure 17) can be used for polynomial multiplication in (5.29).

First: The flip-flops of figure 17 are initially filled with 0's,

Second: Input the sequence u_0, u_1, u_2, u_3 (First-element first) followed by $n - k$ 0's

$$(u_0 \ u_1 \ u_2 \ u_3 \ 0 \ 0 \ 0)$$

one bit every tick to the shift register via the input arrow.

Let us now study the behavior of the circuit at each tick of the clock:

Tick 0: input u_0

Shift registers contents: $(u_0\ \ 0\ \ 0)$

Output $v_0 = u_0 g_0$

Tick 1: input u_1

Shift registers contents: $(u_1\ \ u_0\ \ 0)$

Output $v_1 = u_1 g_0 + u_0 g_1$

Tick 2: input u_2

Shift registers contents: $(u_2\ \ u_1\ \ u_0)$

Output $v_2 = u_2 g_0 + u_1 g_1 + u_0 g_2$

Tick 3: input u_3

Shift registers contents: $(u_3\ \ u_2\ \ u_1)$

Output $v_3 = u_3 g_0 + u_2 g_1 + u_1 g_2 + u_0 g_3$

Tick 4: input 0

Shift registers contents: $(0\ \ u_3\ \ u_2)$

Output $v_4 = u_3 g_1 + u_2 g_2 + u_1 g_3$

Tick 5: input 0

Shift registers contents: $(0\ \ 0\ \ u_3)$

Output $v_5 = u_2 g_3$

Tick 6: input 0

Shift registers contents: $(0\ \ 0\ \ 0)$

Output $v_6 = u_3 g_3$

Hence, the output sequence will be $(v_0\ v_1\ v_2 v_3\ v_4\ v_5\ v_6)$, where the $v_i's$ are defined by equations above.

Now let $g(x) = 1 + x + x^3$ be the generator polynomial of $C(7,4)$ given in (Table 19). Consider the information bits are (1 0 1 1) then the nonsystematic encoder is shown as follows:

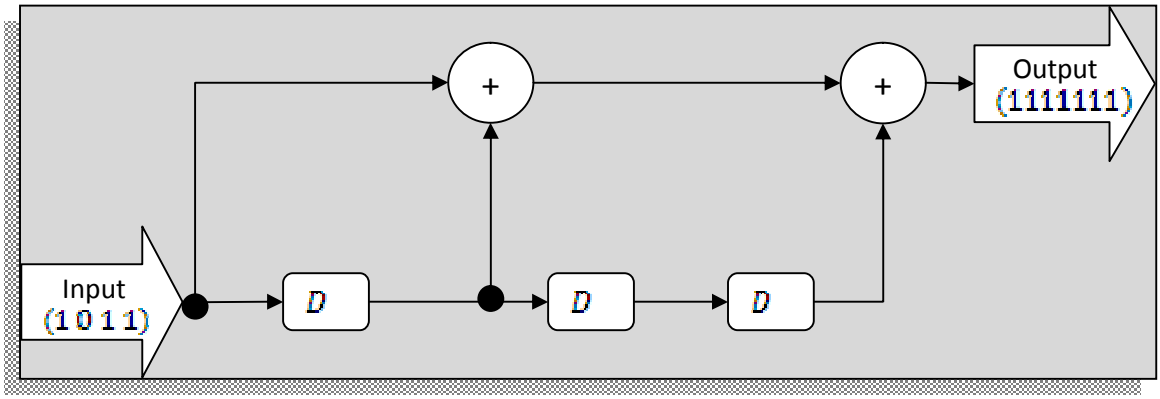


Figure 18: Nonsystematic Encoder for the (7, 4) Cyclic Code with Generator Polynomial $g(x) = 1 + x + x^3$.

First: The flip-flops of figure 18 are initially filled with 0's,

Second: Input the sequence $(1\ 0\ 1\ 1)$ followed by $n - k$ 0's

$(1\ 0\ 1\ 1\ 0\ 0\ 0)$

one bit every tick to the shift register via the input arrow.

Let us now study the behavior of the circuit at each tick of the clock:

Tick 0: input $\mathbf{1}$

Shift registers contents: $(1\ \ 0\ \ 0)$

Output $v_0 = 1$

Tick 1: input 0

Shift registers contents: $(1\ \ 1\ \ 0)$

Output $v_1 = 1$

Tick 2: input $\mathbf{1}$

Shift registers contents: $(1\ \ 0\ \ 1)$

Output $v_2 = 1$

Tick 3: input $\mathbf{1}$

Shift registers contents: $(1\ \ 1\ \ 0)$

Output $v_3 = 1$

Tick 4: input 0

Shift registers contents: $(0\ \ 1\ \ 1)$

Output $v_4 = 1$

Tick 5: input 0

Shift registers contents: $(0\ \ 0\ \ 1)$

Output $v_5 = 1$

Tick 6: input 0

Shift registers contents: $(0\ \ 0\ \ 0)$

Output $v_6 = 1$

Hence, the corresponding codeword will be $(1\ \mathbf{1}\ 1\ 1\ 1\ 1\ \mathbf{1})$.

5.6.2 Systematic Encoder

The encoder of Figure 17 could be simpler, but it is unfortunately not systematic encoder.

However, the idea to design a systematic shift-register encoder for the previous cyclic code is to use the result of section 5.5.2, which says that if $u(x)$ is an information polynomial, then

$$u(x) \rightarrow u(x).x^{n-k} + b(x) \quad mod\ g(x) \tag{5.30}$$

is a systematic encoding rule for a cyclic code with generator polynomial $g(x)$, where $b(x)$ is the remainder polynomial dividing $u(x).x^{n-k}$ by $g(x)$.

The following figure shows the encoding circuit for an (7, 4) cyclic code with generator polynomial $g(x) = g_0 + g_1x + g_2x^2 + g_3x^3$.

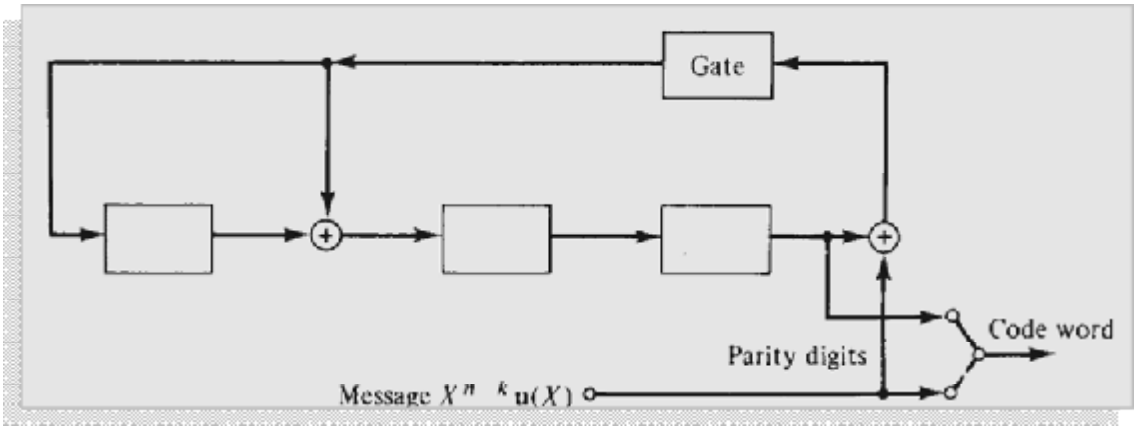


Figure 19: Systematic Shift-Register Encoder for a (7, 4) Cyclic Code with $g(x) = g_0 + g_1x + g_2x^2 + g_3x^3$

In this circuit the flip-flops store the $(n-k)$ parity check digits b_0, b_1, b_2 (the coefficients of $b(x)$) at the last tick.

Note that the right-most symbol of the word is the first symbol to enter the encoder. The gate is turned on until all the information digits have been shifted into the circuit

The encoder operation is carried out as follows:

Step 1: Reset the coefficients of the flip-flops, i.e. $f_i(0) = 0 \ \forall i = 0, 1, 2$

Step 2: The behavior of the circuit at each tick of the clock:

Note that the input to the gate in i^{th} stage is: $f_2(i-1) + u_i$

where f_2 is the stored digit of the second flip-flop in the $i-1$ stage.

Tick 0: Input to the channel: $u_3 = v_6$

Input to the gate: $0 + u_3$

Shift registers contents: $(u_3 g_0 \quad u_3 g_1 \quad u_3 g_2)$

Tick 1: Input to the channel: $u_2 = v_5$

Input to the gate: $u_2 + u_3 g_2$

Shift registers contents:

$(u_2 g_0 + u_3 g_0 g_2 \quad u_2 g_1 + u_3 g_1 g_2 + u_3 g_0 \quad u_3 g_2 + u_2 g_2 + u_3 g_1)$

Tick 2: Input to the channel: $u_1 = v_4$

Input to the gate: $u_1 + u_3 g_1 + u_2 g_2 + u_3 g_2$

Shift registers contents:

$(u_1 g_0 + u_3 g_1 g_0 + u_2 g_2 g_0 + u_3 g_2 g_0$

$u_2 g_0 + u_3 g_0 g_2 + u_1 g_1 + u_3 g_1 + u_2 g_2 g_1 + u_3 g_2 g_1$

$u_2 g_1 + u_3 g_1 g_2 + u_3 g_0 + u_1 g_2 + u_3 g_1 g_2 + u_2 g_2 + u_3 g_2)$

Tick 3: Input to the channel: $u_0 = v_3$

Input to the gate:

$u_0 + u_2 g_1 + u_3 g_1 g_2 + u_3 g_0 + u_1 g_2 + u_3 g_1 g_2 + u_2 g_2 + u_3 g_2$

Shift registers contents:

$(u_0 g_0 + u_2 g_1 g_0 + u_3 g_1 g_2 g_0 + u_3 g_0 + u_1 g_2 g_0 + u_3 g_1 g_2 g_0 + u_2 g_0 g_2$

$+ u_3 g_2 g_0$

$u_1 g_0 + u_3 g_1 g_0 + u_2 g_2 g_0 + u_3 g_2 g_0 + u_0 g_1 + u_2 g_1 + u_3 g_1 g_2 + u_3 g_1 g_0$

$+ u_1 g_2 g_1 + u_3 g_1 g_2 + u_2 g_2 g_1 + u_3 g_2 g_1$

$$u_0g_2 + u_3g_1g_2 + u_2g_2 + u_2g_0 + u_1g_1 + u_3g_0g_1 + u_0g_2 + u_2g_1g_2 + u_3g_1g_2 + u_3g_0g_2 + u_1g_2 + u_3g_1g_2 + u_2g_2 + u_3g_2)$$

Step 3: Break the feedback connection by turning off the gate.

Step 4: Shift the parity-check out and send them into the channel.

The parity-check digits $b_0 = v_0, b_1 = v_1$ and $b_2 = v_2$ are the contents of the shift register in tick 3. In this case,

$$b_0 = u_0g_0 + u_2g_1g_0 + u_3g_1g_2g_0 + u_3g_0 + u_1g_2g_0 + u_3g_1g_2g_0 + u_2g_0g_2 + u_3g_2g_0$$

$$b_1 = u_1g_0 + u_3g_1g_0 + u_2g_2g_0 + u_3g_2g_0 + u_0g_1 + u_2g_1 + u_3g_1g_2 + u_3g_1g_0 + u_1g_2g_1 + u_3g_1g_2 + u_2g_2g_1 + u_3g_2g_1$$

$$b_2 = u_0g_2 + u_3g_1g_2 + u_2g_2 + u_2g_0 + u_1g_1 + u_3g_0g_1 + u_0g_2 + u_2g_1g_2 + u_3g_1g_2 + u_3g_0g_2 + u_1g_2 + u_3g_1g_2 + u_2g_2 + u_3g_2)$$

Hence, the output sequence to the channel will be $(v_0\ v_1\ v_2 v_3\ v_4\ v_5\ v_6)$.

Now let $g(x) = 1 + x + x^3$ be the generator polynomial of $C(7,4)$ given in table 20. Consider the information bits are $(u_0\ u_1\ u_2 u_3)$ then the systematic encoder is shown as follows:

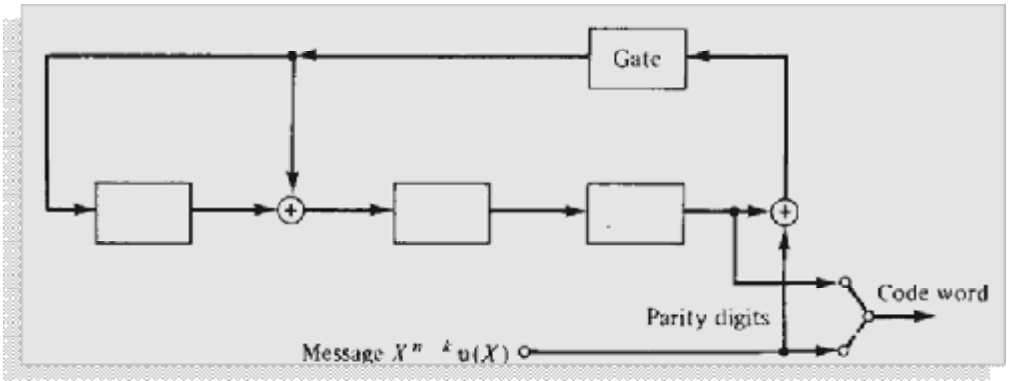


Figure 20: Systematic Shift-Register Encoder for the (7, 4) Cyclic Code with $g(x) = 1 + x + x^3$

Step 1: Reset all the flip-flops.

Step 2:

Tick 0: Input to the channel: $u_3 = v_6$

Input to the gate: $0 + u_3$

Shift registers contents: $(u_3 \quad u_3 \quad 0)$

Tick 1: Input to the channel: $u_2 = v_5$

Input to the gate: $0 + u_2$

Shift registers contents: $(u_2 \quad u_2 + u_3 \quad u_3)$

Tick 2: Input to the channel: $u_1 = v_4$

Input to the gate: $u_1 + u_3$

Shift registers contents: $(u_1 + u_3 \quad u_1 + u_2 + u_3 \quad u_2 + u_3)$

Tick 3: Input to the channel: $u_0 = v_3$

Input to the gate: $u_0 + u_2 + u_3$

Shift registers contents:

$(u_0 + u_2 + u_3 \quad u_0 + u_1 + u_2 \quad u_1 + u_2 + u_3)$

Step 3: Break the feedback connection by turning off the gate.

Step 4: Shift the parity-check out and send them into the channel.

The parity-check digits $b_0 = v_0, b_1 = v_1$ and $b_2 = v_2$ are the contents of the shift register in tick 3. In this case,

$$b_0 = u_0 + u_2 + u_3,$$

$$b_1 = u_0 + u_1 + u_2$$

$$b_2 = u_1 + u_2 + u_3.$$

For example, if the information bits are (1 0 1 1), the corresponding codeword will be (1 0 0 1 0 1 1).

5.7 Cyclic Codes Decoding

Decoding of cyclic codes consists of the same three steps as for decoding linear codes:

1. Syndrome computation,
2. Association of the syndrome to an error pattern,
3. Error correction.

Recall from Chapter 3 for any linear code, we can form a standard array, or we can use the reduced standard array using syndromes. For cyclic codes it is possible to exploit the cyclic structure of the code to decrease the memory requirements.

First we must determine if the received word r is a codeword in C or not using (Theorem 5.5) which is say that an $r(x) \in C$ if and only if $r(x)h(x) \equiv 0 \, mod \, x^n + 1 \Longrightarrow (x^n + 1)\text{divided}r(x)h(x)$

If $r(x) \notin C$ we determine the closest codeword in $C(n,k)$ using the syndrome of $r(x)$ as follows:

Since every valid received code polynomial $r(x)$ must be a multiple of the generator polynomial $g(x)$ of C, then when we divide $r(x)$ by $g(x)$ the remainder is zero exactly when $r(x)$ is a codeword, i.e.

$$r(x) = a(x)g(x) + 0$$

Thus we can employ the division algorithm to obtain a syndrome as follows:

$$r(x) = a(x)g(x) + s(x)$$

where $a(x)$ is the quotient and $s(x)$ is the remainder polynomial having degree less than the degree of $g(x)$:

$s(x) = s_0 + s_1 x + \cdots + s_{n-k-1}x^{n-k-1}$

Thus, to compute the syndrome we can use a circuit such as that in the figure 19 as we will see after the following useful result about cyclic codes and syndromes.

Theorem 5.8: Let $s(x)$ be the syndrome of a received polynomial $r(x) = r_0 + r_1 x + \cdots + r_{n-1}x^{n-1}$. Let $r^{(1)}(x)$ be the polynomial obtained by cyclically right-shifting $r(x)$ and let $s^{(1)}(x)$ denote its syndrome. Then $s^{(1)}(x)$ is the remainder obtained when dividing $xs(x)$ by $g(x)$.

Proof: With $r(x) = r_0 + r_1 x + \cdots + r_{n-1}x^{n-1}$ the cyclic shift $r^{(1)}(x)$ is $r^{(1)}(x) = r_{n-1} + r_0 x + \cdots + r_{n-2}x^{n-1}$,

which can be written as

$$xr(x) = r_{n-1}(x^n + 1) + r^{(1)}(x) \qquad (5.31)$$

It follows from (Theorem 5.1).

Rearranging (5.31), we have

$$r^{(1)}(x) = r_{n-1}(x^n + 1) + xr(x) \tag{5.32}$$

Dividing both sides of (5.32) by $g(x)$ and using the fact that $x^n + 1 = g(x).h(x)$ and $r(x) = a(x)g(x) + s(x)$, we obtain

$$a^{(1)}(x)g(x) + s^{(1)}(x) = r_{n-1}g(x)h(x) + x(a(x)g(x) + s(x)) \tag{5.33}$$

Where $s^{(1)}(x)$ is the remainder resulting from dividing $r^{(1)}(x)$ by $g(x)$, which is the syndrome of $r^{(1)}(x)$.

Rearranging (5.33), we obtain the following relationship between $s^{(1)}(x)$ and $xs(x)$:

$$xs(x) = \left[a^{(1)}(x) + r_{n-1}h(x) + xa(x)\right]g(x) + s^{(1)}(x) \tag{5.34}$$

Thus $s^{(1)}(x)$ is the remainder from dividing $xs(x)$ by $g(x)$

Hence, $s^{(1)}(x)$ is the syndrome of $r^{(1)}(x)$. ■

By induction, the syndrome $s^{(i)}(x)$ that corresponds to cyclically shifting $r(x)$ i times to produce $r^{(i)}(x)$ is the remainder of $x^i s(x)$ when divided by $g(x)$.

Example 5.14: For the (7, 4) cyclic code with generator $g(x) = x^3 + x + 1$, let $r = (0\ 1\ 1\ 0\ 1\ 1\ 1)$ be the received word. The syndrome of $r(x)$ can be computed from dividing $r(x) = x + x^2 + x^4 + x^5 + x^6$ by $g(x) = 1 + x + x^3$

If $1 + x + x^3 = 0 \Longrightarrow x^3 = 1 + x$

Then $r(x) = x + x^2 + x^4 + x^5 + x^6$

$$= x + x^2 + x(1+x) + x^2(1+x) + (1+x)^2$$

$$= x + x^2 + x + x^2 + x^2 + 1 + x + 1 + x^2 = x$$

$\overset{then}{\Longrightarrow}$ The remainder is the syndrome of r

$s = x \Longrightarrow s = (0\ 1\ 0)$ is the syndrome.

Then the cyclic shifts of $r(x)$ and their corresponding syndromes are shown in the following table:

Table 21: Corresponding Syndromes of the Cyclic Shifts of $r(x)$

Polynomial	Syndrome
$r(x) = x + x^2 + x^4 + x^5 + x^6$	$s(x) = x$
$r^{(1)}(x) = 1 + x^2 + x^3 + x^5 + x^6$	$s^{(1)}(x) = x^2$
$r^{(2)}(x) = 1 + x + x^3 + x^4 + x^6$	$s^{(2)}(x) = 1 + x$
$r^{(3)}(x) = 1 + x + x^2 + x^4 + x^5$	$s^{(3)}(x) = x + x^2$
$r^{(4)}(x) = x + x^2 + x^3 + x^5 + x^6$	$s^{(4)}(x) = 1 + x + x^2$
$r^{(5)}(x) = 1 + x^2 + x^3 + x^4 + x^6$	$s^{(5)}(x) = 1 + x^2$
$r^{(6)}(x) = 1 + x + x^3 + x^4 + x^5$	$s^{(6)}(x) = 1$

The syndrome computation can be accomplished with a division circuit as shown in the following figure:

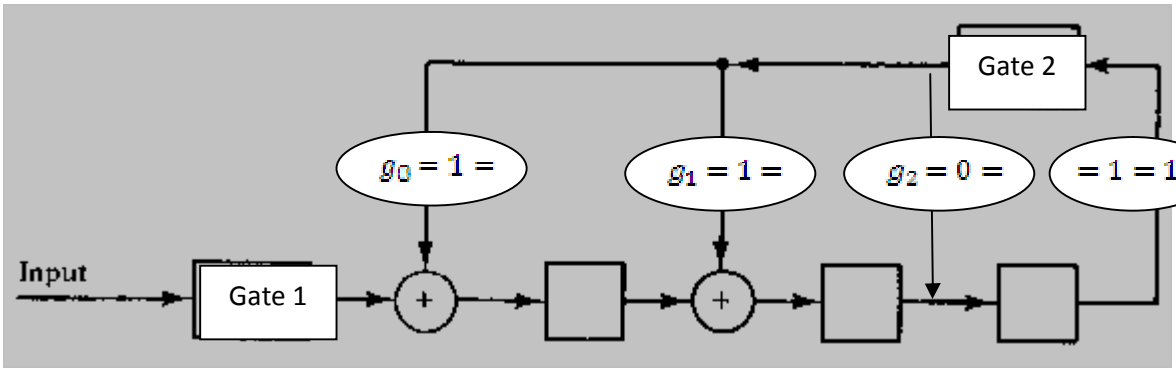


Figure 21: Syndrome Circuit for the (7, 4) Cyclic Code Generated by $g(x) = 1 + x + x^3$

This circuit consists of $n-k$ stage and it is shifting the received word $r = (r_0\ r_1\ r_2\ r_3\ r_4\ r_5\ r_6)$ from the right end.

The behavior of the circuit at each tick of the clock is;

Note that the received polynomial is shifted into the register with all stages initially set to 0.

Tick 0: Input to the gate 1: r_6

Shift registers contents: $(r_6 \quad 0 \quad 0)$

Input to the gate 2: 0

Tick 1: Input to the gate 1: r_5

Shift registers contents: $(r_5 \quad r_6 \quad 0)$

Input to the gate 2: 0

Tick 2: Input to the gate 1: r_4

Shift registers contents: $(r_4 \quad r_5 \quad r_6)$

Input to the gate 2: r_6

Tick 3: Input to the gate 1: r_3

Shift registers contents: $(r_3 + r_6 \quad r_4 + r_6 \quad r_5)$

Input to the gate 2: r_5

Tick 4: Input to the gate 1: r_2

Shift register contents: $(r_2 + r_5 \quad r_3 + r_5 + r_6 \quad r_4 + r_6)$

Input to the gate 2: $r_4 + r_6$

Tick 5: Input to the gage 1: r_1

Shift register content:

$(r_1 + r_4 + r_6 \quad r_2 + r_4 + r_5 + r_6 \quad r_3 + r_5 + r_6)$

Input to the gate 2: $r_3 + r_5 + r_6$

Tick 6: Input to the gate 1: r_0

Shift register content:

$$(r_6 + r_5 + r_3 + r_0 \quad r_5 + r_4 + r_3 + r_1 \quad r_6 + r_5 + r_4 + r_2)$$

As soon as the entire $r(x)$ has been shifted into the register, break the feedback connection by turning off gate 1 where the syndrome $s(x) = s_0 + s_1 x + s_2 x^2$ in its registers in tick 6. In this case,

$s_0 = r_6 + r_5 + r_3 + r_0$, $s_1 = r_5 + r_4 + r_3 + r_1$ & $s_2 = r_6 + r_5 + r_4 + r_2$.

For example, if the received word is $(0\ 0\ 1\ 0\ 1\ 1\ 0)$, the corresponding syndrome will be (1 0 1).

Now after the gate 1 is closed the system will be shifted 6 or more times. The registers contain successively the syndromes $s^{(i)}(x)$ corresponding to the cyclically shifted polynomials $r^{(i)}(x)$, which is showed in (Table 5.21).

That operations can be shown in the following steps:

Tick 7: Input to the gate 2: $r_2 + r_4 + r_5 + r_6$

Output: the contents of the shift register:

$$s^{(1)}(x) = (r_2 + r_4 + r_5 + r_6 \quad r_0 + r_2 + r_3 + r_4 \quad r_1 + r_3 + r_4 + r_5)$$

Tick 8: Input to the gate 2: $r_1 + r_3 + r_4 + r_5$

Output: the contents of the shift register:

$$s^{(2)}(x) = (r_1 + r_3 + r_4 + r_5 \quad r_1 + r_2 + r_3 + r_6 \quad r_0 + r_2 + r_3 + r_4)$$

Tick 9: Input to the gate 2: $r_0 + r_2 + r_3 + r_4$

Output: the contents of the shift register:

$$s^{(3)}(x) = (r_0 + r_2 + r_3 + r_4 \quad r_0 + r_1 + r_2 + r_5 \quad r_1 + r_2 + r_3 + r_6)$$

Tick 10: Input to the gate 2: $r_1 + r_2 + r_3 + r_6$

Output: the contents of the shift register:

$$s^{(4)}(x) = (r_1 + r_2 + r_3 + r_6 \quad r_0 + r_1 + r_4 + r_6 \quad r_0 + r_1 + r_2 + r_5)$$

Tick 11: Input to the gate 2: $r_0 + r_1 + r_2 + r_5$

Output: the contents of the shift register:

$$s^{(5)}(x) = (r_0 + r_1 + r_2 + r_5 \quad r_0 + r_3 + r_5 + r_6 \quad r_0 + r_1 + r_4 + r_6)$$

Tick 12: Input to the gate 2: $r_0 + r_1 + r_4 + r_6$

Output: the contents of the shift register:

$$s^{(6)}(x) = (r_0 + r_1 + r_4 + r_6 \quad r_2 + r_4 + r_5 + r_6 \quad r_0 + r_3 + r_5 + r_6)$$

For instance, the following table shown how we can computing the syndrome and its cyclic shifts for $r = (0\ 1\ 1\ 0\ 1\ 1\ 1)$.

Table 22: Computing the Syndrome and its Cyclic Shifts

Clock	Input	Registers	syndrome
Initial:		0 0 0	
1	1	1 0 0	
2	1	1 1 0	
3	1	1 1 1	
4	0	1 0 1	
5	1	0 0 0	
6	1	1 0 0	
7	0	0 1 0	
……(turn off gate)			
8		0 0 1	$s^{(1)}(x) = x^2$
9		1 1 0	$s^{(2)}(x) = 1 + x$

10	0 1 1	$s^{(3)} = x + x^2$
11	1 1 1	$s^{(4)}(x) = 1 + x + x^4$
12	1 0 1	$s^{(5)} = 1 + x^2$
13	1 0 0	$s^{(6)}(x) = 1$

Now, Let $r(x)$ be the received polynomial with the syndrome $s(x)$, i.e.

$$r(x) = a(x)g(x) + s(x) \tag{5.35}$$

and let $e(x) = e_0 + e_1 x + \cdots + e_{n-1}x^{n-1}$ be the error pattern. Then the transmitted codeword $v(x)$ is:

$$v(x) = r(x) + e(x) \tag{5.36}$$

Since $v(x)$ is a multiple of the generator polynomial $g(x)$, i.e. $v(x) = b(x)g(x)$, combining (5.35) and (5.36), we have the following relationship between the error pattern and the syndrome:

$$e(x) = [a(x) + b(x)]g(x) + s(x) \tag{5.37}$$

This shows that the syndrome is equal to the remainder resulting from dividing $e(x)$ by $g(x)$.

However, the error pattern $e(x)$ is unknown to the decoder. Therefore, the decoder has to estimate $e(x)$ based on the syndrome $s(x)$. If $e(x)$ is a coset leader in the standard array and if table-lookup decoding is used, $e(x)$ can be correctly determined from the syndrome.

From (5.37), we see that $s(x)$ is identical to zero if and only if either $e(x) = 0$ or it is identical to a codeword, in other words if $e(x)$ is an undetectable error patterns.

Remark 5.6: The minimum distance of the cyclic code $C(n,k)$ is equal to the minimum weight of $C(n,k)$ which is equal to the degree of the minimum polynomial $g(x)$ of $C(n,k)$ that is $n-k$.

Remark 5.7: The cyclic code $C(n,k)$ is capable of correcting up to $t = \left[\frac{d_{min}-1}{2}\right] = \left[\frac{(n-k)-1}{2}\right]$ errors made by channel.

Example 5.15: Consider again the decoder for the cyclic code with generator polynomial $g(x) = 1 + x + x^3$. The following table-lookup decoding shows the error vectors and their corresponding syndrome vectors and polynomials. The code has $2^{7-4} = 2^3 = 8$ cosets and, therefore, there are eight correctable error patterns (including the zero word). Since the minimum distance of the code is 3, it is capable of correcting all the error patterns of weight 1 or 0. Hence, all the 7-tuples of weight 1 or 0 can be used as coset leaders. There are $\binom{7}{0} + \binom{7}{1} = 8$ coset leaders, which is shown in the following table.

Table 23: Decoding Table for the (7, 4) Cyclic Code generated by $g(x) = 1 + x + x^3$

Syndrome s	Syndrome polynomial $s(x)$	Error e	Error polynomial $e(x)$
000	0	0000000	0
100	1	1000000	1
010	x	0100000	x
001	x^2	0010000	x^2
110	$1 + x$	0001000	x^3
011	$x + x^2$	0000100	x^4
111	$1 + x + x^2$	0000010	x^5
101	$1 + x^2$	0000001	x^6

Let $r = (0\ 1\ 1\ 0\ 1\ 1\ 1)$ be the received word. The syndrome of r showed in previous example where $s = (0\ 1\ 0)$, then from this table we recognize that the received polynomial $r(x)$ has an error in the second bit.

Thus the transmitted codeword is:

$$v = r + e = (0\ 1\ 1\ 0\ 1\ 1\ 1) + (0\ 1\ 0\ 0\ 0\ 0\ 0) = (0\ 0\ 1\ 0\ 1\ 1\ 1).$$

References

1. Attarian Ad., Hutzel An. And Neal Ry. **Algebraic Coding Theory**. 2006. 12P.
2. Berlekamp E. **A Survey of Algebraic Coding Theory**. No. 28. New York: Wien; 1970. 38P.
3. Bhattacharya P., Jain S. and Nagpaul S. **Basic Abstract Algebra**. 2nd ed. Cambridge University; 1994. 486P.
4. Blahut R. **Algebraic Codes for Data Transmission**. United Kingdom: Cambridge University Press; 2003. 482p.
5. Doran R., Ismail M., Lam T., Lutwak E. and Spigler R. **Encyclopedia of Mathematics and its Applications**. 2nd ed. Cambridge University Press; 2002. 205.
6. Hall J. **Notes on Coding Theory**. United State America: Michigan State University. 2003. 10P.
7. Hamming R. **Error Detecting and Error Correcting Codes**. Bell Syst. Tech. J., 29. 1950; 147-160.
8. Han Y. **Introduction to Binary Linear Block Codes**. National Taipei University. Taiwan. 97P.
9. Kolman B. **Introductory Linear Algebra**: with Applications. 3rd ed. United States of America: Prentice Hall; 1997. 608P.
10. Kabatiansky G, Krouk E, Semenov S. **Error Correcting Coding and Security for Data Networks**. John Wiley & Sons, Ltd; 2005. 278p
11. Lemmermeyer F. **Error Correcting Codes**. 2005. 100P.

12. Lint J. **Graduate Texts in Mathematics: Introduction to Coding Theory**. 3rd ed. Netherlands: Springer; 1999. 227p.
13. Ling Sa. And Xing CH. **Coding Theory: A First Course**. Cambridge University. 2004. 222P.
14. Lin SH, Costello DA. **Error control coding: Fundamentals and Applications**. United States of America: Prentice-Hall; 1983. 603p.
15. Moon T. **Error correction coding: Mathematical Methods and Algorithms**. United States of America: John Wiley and Sons; 2005. 756p
16. Roweis S. **Equivalent Codes & Systematic Forms**. Lecture. 2006.
17. Valenti M. **The Evolution of Error Control Coding**. Member, IEEE.

Appendix A

1. An Elementary Row Operation

Let A be a matrix over $GF(2)$; an elementary row operation performed on A is any one of the following three operations:

(i) Interchanging two rows,

(ii) Multiplying a row by a nonzero scalar,

(iii) Replacing a row by its sum with the scalar multiple of another row.

2. Reduced Row Echelon Form (RREF)

An mxn matrix is said to be in reduced row echelon form (RREF) when it satisfies the following properties:

(i) All rows consisting entirely of zeros, if any, are at the bottom of the matrix.

(ii) Reading from left to right, the first nonzero entry in each row that does not consist entirely of zeros is a $\mathbf{1}$, called the *leading entry* of its row.

(iii) If rows i and $i+1$ are two successive rows that do not consist entirely of zeros, then the leading entry of row $i+1$ is to the right of the leading entry of row i.

(iv) If a column contains a leading entry of some row, then all other entries in that column are zero.

3. Solutions of Linear Systems of Equations

A linear system of the form

$$v_{11}c_1 + v_{12}c_2 + \cdots + v_{1k}c_k = 0$$
$$v_{21}c_1 + v_{22}c_2 + \cdots + v_{2k}c_k = 0$$

$$\vdots \qquad \text{(eq. 1)}$$
$$v_{n1}c_1 + v_{n2}c_2 + \cdots + v_{nk}c_k = 0$$

is called a homogeneous system. We can also write (eq. 1) in matrix form as $A.X = 0$ (eq. 2)

Where: $A_{n\times k} = \begin{bmatrix} v_{11} v_{12} \ldots v_{1k} \\ v_{21} v_{22} \ldots v_{2k} \\ \vdots \quad \vdots \quad \vdots \\ v_{n1} \quad v_{n2} \quad v_{nk} \end{bmatrix}$ and $X_{k\times 1} = \begin{bmatrix} c_1 \\ c_2 \\ \vdots \\ c_k \end{bmatrix}$

The augmented matrix of this system,

$$\left(\begin{array}{ccc|c} v_{11} v_{12} \ldots v_{1k} & 0 \\ v_{21} v_{22} \ldots v_{2k} & 0 \\ \vdots \quad \vdots \quad \vdots & \vdots \\ v_{n1} \quad v_{n2} \quad v_{nk} & 0 \end{array}\right)$$

Using reduced row echelon form (RREF) The solution is:

$$c_1 = c_2 = \cdots = c_k = 0$$

To the homogeneous system (eq. 2) is called the trivial solution. A solution $c_1, c_2, \ldots, c_k$ to a homogeneous system in which not all the x_i are zero is called a nontrivial solution.

4. Linearly Dependent & Linearly Independent

The procedure to determine if the vectors $v_1, v_2, \ldots, v_k$ are linearly dependent or linearly independent is as follows:

Step 1: Form Equation,

$c_1v_1 + c_2v_2 + \ldots + c_kv_k = 0$, (eq. 3)

which leads to a homogeneous system.

Step 2: Construct the augmented matrix associated with the homogeneous system of (eq. 1). And transform it to reduced row echelon form.

Step 3: If the homogeneous system has only the trivial solution, then the given vectors are linearly independent; if it has a nontrivial solution, then the vectors are linearly dependent.

5. Basis

Let $S = \{v_1, v_2, \ldots, v_k\}$ be a set of nonzero vectors in a vector space V_n .

The procedure for finding a subset of S that is a basis for span S is as follows:

Step 1: Form (e.q. 1),

$$c_1 v_1 + c_2 v_2 + \ldots + c_k v_k = 0,$$

which leads to a homogeneous system.

Step 2: Construct the augmented matrix associated with the homogeneous system of equation (1). And transform it to reduced row echelon form.

Step 3: The vectors corresponding to the columns containing the leading $1's$ form a basis for span S.

Appendix B

Algorithm (1)

Input: A nonempty subset B of V_n.

Output: A basis for $C =< B >$, the linear code generated by a non empty set B.

Description: Form the matrix A whose columns are the nonzero codewords in C.

Use elementary row operations to put A in REF and locate the leading columns in the REF. Then the original columns of A corresponding to these leading columns form a basis B for C.

Algorithm (2)

Input: A nonempty subset B of V_n.

Output: A basis for the dual code $C^{\perp}$, where $C =< B >$.

Description: Form the $2^k \times n$ matrix A whose rows are all codewords in C.

Use elementary row operations to place A in RREF. Let G be the $k \times n$ submatrix of A consisting of all the nonzero rows of the RREF: $A = \binom{G}{0}$

(Here, 0 denotes the $(2^k - k) \times n$ zero submatrix).

The matrix G contains k leading columns. Permute the columns of G to form $\bar{G} = G.P = (X|I_k)$ where I_k denotes the $k \times k$ identity matrix. Form a matrix $\bar{H}$ as follows: $\bar{H} = (I_{n-k}|-X^T)$ where X^T denotes the transpose of X. Apply the inverse of the permutation applied to the columns of G to the columns of $\bar{H}$ to form $H = \bar{H}.P^T$. Then the rows of H form a basis for $C^{\perp}$.

جامعة النجاح الوطنية
عمادة كلية الدراسات العليا

اكتشاف الأخطاء وتصحيحها باستخدام شيفرات هامنج والشيفرات الحلقية

إعداد

نِعم هاشم إبراهيم إبراهيم

إشراف

د. "محمد عثمان" عمران

قدمت هذه الأطروحة استكمالا لمتطلبات درجة الماجستير في الرياضيات بكلية الدراسات العليا
في جامعة النجاح الوطنية في نابلس، فلسطين
2009

اكتشاف الأخطاء وتصحيحها باستخدام شيفرات هامنج والشيفرات الحلقية

إعداد

نِعم هاشم إبراهيم إبرِاهيم

إشراف

د. "محمد عثمان" عمران

الملخص

تناقش الرسالة كيفية تشفير الرسائل القادمة من المرسل عبر قنوات اتصال ثنائية إلى شيفرات جماعية خطية مبنية على نظام شيفرات هامنج والشيفرات الحلقية.

ثم التحقق من مدى صحة الشيفرات المستلمة من قبل المستلم والبحث عن كيفية لتصحيح الأخطاء (إن وجدت) الناجمة عن قنوات الاتصال.

وأخيرا فك تلك الشيفرات المعدلة وإعادتها إلى الشكل الذي أرسلت عليه من قبل المرسل.