

# Runlength Limited Codes with Error Correction Capabilities

by

Sajid Hussain

A Thesis Presented to the

FACULTY OF THE COLLEGE OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**MASTER OF SCIENCE**

In

**ELECTRICAL ENGINEERING**

March, 1995

## **INFORMATION TO USERS**

**This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.**

**The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.**

**In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.**

**Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book.**

**Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.**

# **UMI**

A Bell & Howell Information Company  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
313/761-4700 800/521-0600





**RUNLENGTH LIMITED CODES WITH  
ERROR CORRECTION CAPABILITES**

BY

***SAJID HUSSAIN***

A Thesis Presented to the  
FACULTY OF THE COLLEGE OF GRADUATE STUDIES  
KING FAHD UNIVERSITY OF PETROLEUM & MINERALS  
DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the  
Requirements for the Degree of

**MASTER OF SCIENCE**  
In  
**ELECTRICAL ENGINEERING**

**March 1995**

**UMI Number: 1375331**

---

**UMI Microform 1375331**

**Copyright 1995, by UMI Company. All rights reserved.**

**This microform edition is protected against unauthorized  
copying under Title 17, United States Code.**

---

**UMI**

**300 North Zeeb Road  
Ann Arbor, MI 48103**

**KING FAHD UNIVERSITY OF PETROLEUM AND MINERALS**  
**DHAHRAN, SAUDI ARABIA**  
**COLLEGE OF GRADUATE STUDIES**

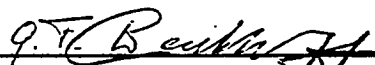
*This thesis, written by*

**Sajid Hussain**

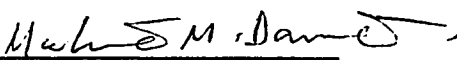
*under the direction of his Thesis Advisor, and approved by his Thesis committee, has  
been presented to and accepted by the Dean, College of Graduate Studies, in partial  
fulfillment of the requirements for the degree of*

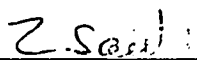
**MASTER OF SCIENCE IN  
ELECTRICAL ENGINEERING**

**Thesis Committee :**


  
Dr. Gerhard F. Beckhoff (Chairman)

  
Dr. Abdul-Rahman Kalaf Al-Ali (Member)

  
Dr. Mahmoud M. Dawoud (Member)

  
Dr. Zeini Jamal Al-Saati (Member)

  
Department Chairman

  
Dean, College of Graduate Studies

Date: 27-6-95



Dedicated to

my parents

&

and Mohni and Saqib

whose prayers, guidance, and inspiration led to

this accomplishment

## ACKNOWLEDGEMENT

In the name of Allah, Most Gracious, Most Merciful. Read in the name of thy Lord and Cherisher, Who created. Created man from a {leech-like} clot. Read and thy Lord is Most Bountiful. He Who taught {the use of} the pen. Taught man that which he knew not. Nay, but man doth transgress all bounds. In that he looketh upon himself as self-sufficient. Verily, to thy Lord is the return {of all}.

(The Holy Quran, Surah 96)

First and foremost, all praise to Allah, *subhanahu-wa-ta'ala*, the Almighty. Who gave me an opportunity, courage and patience to carry out this work. I feel privileged to glorify His name in the sincerest way through this small accomplishment. I seek His mercy, favor, and forgiveness. And I ask Him to accept my little effort. May He, *subhanahu-wa-ta-Aaala*, guide us and the whole humanity to the right path (*Aameen*).

Acknowledgement is due to King Fahd University of Petroleum & Minerals for providing support to this research work.



I am indebted to my thesis chairman, Dr. Gerhard F. Beckhoff for his help and advice. I acknowledge him for his valuable time, encouragement and guidance especially during the early stages of this work and my MS studies. Working with him was indeed a learning experience.

I am grateful to my thesis committee member, Dr. Abdul Rahman Kalaf Al-Ali for his deep interest, constructive criticism and stimulating discussions during the course of this work. Thanks are also due to the my thesis committee members, Dr. Mahmoud M. Dawoud and Dr. Zeini Jamal Al-Saati for their comments and critical review of the thesis. I am thankful to Dr. Kousa Maan for his guidance and help during research. I am very grateful to Dr. Andy Popplewell (U.K.) for his cooperation in developing software.

I am thankful to the department chairman, Dr. Al Abdallah Al Shehri and other faculty members for their cooperation.

I am thankful to my fellow graduate students and all my friends on the campus especially Asif Sayani, Mohammad Ahsan, Amir Farouqi, Osama Abdul Wahab, Syed Zaka Ahmad, Mohammad Rashid, Shahzada Shuja and Abbas Qureshi who provided a wonderful company.

Last but not the least, thanks are due to the members of my family for their

emotional and moral support throughout my academic career. No personal development could ever take place without the proper guidance of parents. This work is dedicated to my parents for having taken pains to fulfill my academic pursuits and shaping my personality. They taught me the fundamental concept of life, "Tough times never last, tough people do". I acknowledge with gratitude, the affection and encouragement of my family members that helped me overcome homesickness and concentrate on my work.

# Contents

ACKNOWLEDGEMENT	i
LIST OF TABLES	viii
LIST OF FIGURES	ix
ABSTRACT (ENGLISH)	
ABSTRACT (ARABIC)	i
1 INTRODUCTION	1
1.1 Coding . . . . .	1
1.2 Channel Constraints . . . . .	2
1.3 Inter Symbol Interference . . . . .	6
1.4 Physical Attributes . . . . .	7
1.5 Sofic Systems . . . . .	8
1.6 Runlength-limited Codes . . . . .	9

1.7	dk-limited binary codes . . . . .	10
<b>2</b>	<b>LITERATURE SURVEY</b>	<b>16</b>
2.1	Runlength-Limited Error Control Codes . . . . .	16
2.2	DC-free Sequences (Bounded Running Sequences) . . . . .	20
2.3	Running Digital Sum (RDS) Constrained . . . . .	21
2.4	Problem Definition . . . . .	23
<b>3</b>	<b>GENERAL PROPERTIES OF TRANSPARENT CODES</b>	<b>25</b>
3.1	Galois Field . . . . .	25
3.2	Linear Codes . . . . .	26
3.3	Quasi-Cyclic Codes . . . . .	28
3.4	Transparent Codes . . . . .	31
3.4.1	Logical Inverse . . . . .	31
3.5	Reflective Codes . . . . .	35
3.6	Runlength . . . . .	36
<b>4</b>	<b>TRANSPARENT CODES</b>	<b>40</b>
4.1	Hamming Codes . . . . .	40
4.2	Repetition Codes . . . . .	48
4.3	Perfect Codes . . . . .	48
<b>5</b>	<b>MODIFIED CODES</b>	<b>53</b>

5.1	Modification of Linear Codes . . . . .	53
5.2	Modification Vector . . . . .	56
5.3	Examples . . . . .	62
5.4	Generator Matrix Transformation . . . . .	72
<b>6</b>	<b>REFLECTIVE CODES</b>	<b>83</b>
6.1	Parity-check Matrix Construction . . . . .	83
6.2	Modification Vector . . . . .	86
6.3	Generator Matrix . . . . .	87
6.3.1	Generator Matrix of Subclass of Reflective Codes . . . . .	88
6.4	Software . . . . .	88
6.5	Examples . . . . .	91
6.5.1	Example #1 . . . . .	91
6.5.2	Example #2 . . . . .	92
6.5.3	Example # 3 . . . . .	94
<b>7</b>	<b>WEIGHT DISTRIBUTION</b>	<b>96</b>
7.1	Weight Enumerator . . . . .	96
7.2	Weight Distribution of Maximum Distance Separable (MDS) codes . . . . .	97
7.3	Weight Distribution of Hamming-Codes . . . . .	98
7.4	Examples . . . . .	102
7.5	Weight Distribution of Subclass of Reflective Codes . . . . .	106

	vii
7.6 Software . . . . .	110
<b>8 CONCLUSIONS AND FUTURE RESEARCH</b>	<b>112</b>
<b>REFERENCES</b>	<b>114</b>
<b>VITAE</b>	<b>120</b>

# List of Tables

1.1	Precoding . . . . .	12
1.2	Various codes with runlength parameters $d$ and $k$ . . . . .	14
2.1	Set of Error Types in Different Channel Models . . . . .	19
3.1	$(6,3)$ 2-cyclic code . . . . .	30
3.2	Symbols of $GF(2^3)$ expressed in binary form . . . . .	32
4.1	Weight Distribution of Golay Code $(n,k,d) = (23,12,7)$ . . . . .	51
7.1	Weight Distribution of Hamming Code $(n,k) = (7,4)$ . . . . .	100
7.2	Weight Distribution of Hamming Code $(n,k) = (15,11)$ . . . . .	101
7.3	Weight Distribution $(n,k) = (20,9)$ . . . . .	107
7.4	Weight Distribution $(n,k) = (20,11)$ . . . . .	108
7.5	Weight Distribution $(n,k) = (32,15)$ . . . . .	109
7.6	Weight Distribution $(n,k) = (32,19)$ . . . . .	111

# List of Figures

1.1	Read-out signal for six pit edges on the disc . . . . .	5
4.1	Properties of Hamming Codes . . . . .	41
4.2	Parity-check matrix of the Hamming code (7,4) . . . . .	43
4.3	Parity-check matrix of the Hamming code (15,11) . . . . .	44
4.4	Parity-check matrix of the dual Hamming code (7,4) . . . . .	46
4.5	Parity-check matrix of the dual Hamming code (15,11) . . . . .	47
5.1	Block diagram of a system employing runlength limited ECCs . . . . .	55
5.2	Algorithm for suitable modification vector . . . . .	59
6.1	Flowchart for obtaining G matrix . . . . .	90
7.1	Weight distribution of (12,5) PO code . . . . .	103
7.2	Weight distribution of (12,6) PO code . . . . .	105



## ABSTRACT

**Name:** Sajid Hussain  
**Title:** Runlength-limited Codes with  
Error Correction Capabilities  
**Major Field:** Electrical Engineering  
**Date of Degree:** March, 1995

*Runlength limited codes with error correction capabilities are investigated. For all storage devices there are some constraints on the runlengths of the data. We proposed an algorithm for binary extension fields which modify the codewords in such a way that runlength constraints are also satisfied. This modification preserves the error correction capability of the code, and thus we obtain extra benefit that runlength constraints are also satisfied in addition to error correction. In transparent codes, the complement of every codeword is also a valid codeword. These transparent codes may give rise to infinite runlengths. Many known codes are found to be transparent. Some general properties of these transparent codes were investigated. Weight distribution of subclass of codes proposed by Popplewell and O'Reilly were also obtained.*

Master of Science Degree  
King Fahd University of Petroleum and Minerals  
Dhahran, Saudi Arabia  
March, 1995

## ملخص الرسالة

الاسم : ساجد حسين  
عنوان الرسالة : الرموز ذات التسلسل المحدود ، والقادرة على تصحيح الأخطاء  
التخصص : الهندسة الكهربائية  
تاريخ الرسالة : مارس ١٩٩٥ م

هذا البحث يُعنى بدراسة الرموز ذات التسلسل المحدود . من المعلوم أن هناك قيوداً على طول السلسلة للرموز المستخدمة في أجهزة تخزين المعلومات . هذا البحث يقترح برنامجاً لتحسين الرموز ( في مجالات موسعة عن المجال الثنائي ) بحيث يحقق القيود المعطاة لطول التسلسل .

من المعلوم أنه في الشفرة الشفافة ، فإن كل العكس الثنائي لكل كلمة مشفرة هي كلمة مشفرة أيضاً . هذه الشفرة الشفافة تحتوي سلاسل لامتتية . كثير من الشفرات العروفة وجد أنها شفافة أيضاً . وفي هذا البحث يدرس الخواص العامة لهذه الشفرات الشفافة . كما وصل هذا البحث إلى إيجاد التوزيع الوزني لمجموعة جزئية من الشفرات بناء على الطريقة المقترحة من بوبل ول واوريلي .

درجة الماجستير في العلوم  
جامعة الملك فهد للبترول والمعادن  
الظهران ، المملكة العربية السعودية  
مارس ١٩٩٥ م

# Chapter 1

## INTRODUCTION

Advances in coding plays an important role in the astounding increase in capacity of today's digital storage. The advances in materials, manufacturing techniques, electronics and mechanical design make compact discs, digital audiotape, floppy disks, video disks, and hard disks possible. But the usage of storage capacity close to their theoretical maximum is made possible by coding.

### 1.1 Coding

Coding techniques are used in communication systems to improve the reliability of the communication channel. The reliability is commonly expressed in terms of the probability of receiving the wrong information. Information theory partitions the coding problem into two main categories: source and channel coding. Source

coding is a technique to reduce the source symbol rate by removing the redundancy in the source signal. Channel coding is the technique of realizing high transmission reliability despite shortcomings of the channel, while making efficient use of the channel capacity.

Coding is a set of rules for assigning, to a source sequence, another sequence that is recorded. The aim of this transformation is to improve the reliability or efficiency of the recording channel.

In recorder systems, channel coding consists of two steps, finding

1. an error correction code,
2. a recording code.

Error correction control is performed by adding extra symbols to the message. Reed-Solomon codes are quite suitable for recording applications because they can combat combinations of random as well as burst errors. Recording code converts the input bit stream to a waveform suitable for the specific recorder requirements.

## 1.2 Channel Constraints

The special attributes that the recorded sequences should have to render it compatible with the physical characteristics of the available transmission channel are called channel constraints.

Long sequences of like symbols easily foil the timing recovery or the adaptive equalization. It seems quite reasonable to try to protect against such vexatious cases by removing them from the input. A coding step in which particular sequences are forbidden to minimize the effect of worst-case patterns is a good example of a recording code. The channel constraints considered here are deterministic by nature and are always in force. A typical example of a channel constraint and its description in time-domain terms is that of sequences containing restricted run length of like symbols. Channel constraints can also be described in frequency-domain terms. Common magnetic recorders do not respond to low-frequency signals so that in order to minimize distortion of the retrieved data, one usually eliminates low frequency components in the recorded data by a coding step.

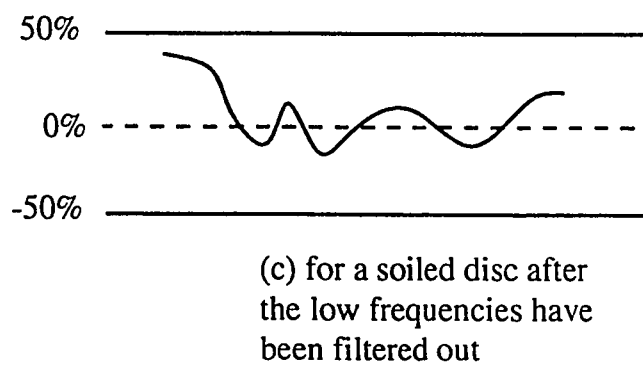
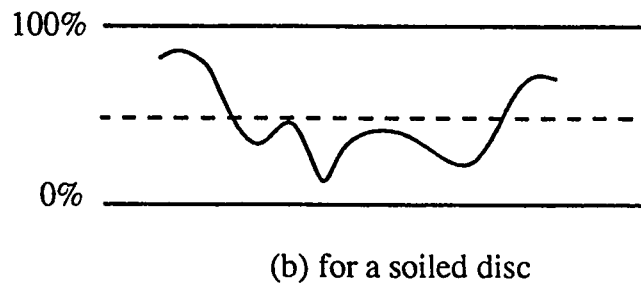
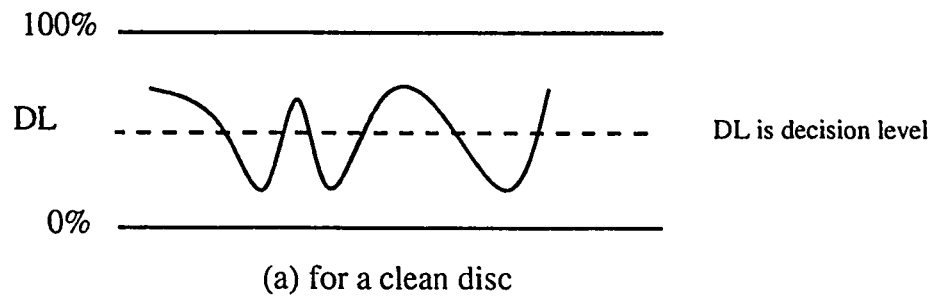
Scrambling is often advocated to eliminate 'worst-case' effects in the data. Scramblers use pseudo-random sequences to randomize the statistics of the data, making them look more like a stationary sequence. There are, however, input patterns, for which scramblers will fail, since any technique that performs a one-to-one mapping between input and output data by necessity does not remove error-prone sequences and remains vulnerable to problematic inputs. Recording codes are also used to include position information for servo systems and timing information for clock recovery.

The digital symbols can be represented by physical quantities in many ways. Most digital recording systems currently in use are binary and synchronous, which

means that in each symbol time interval, or time slot, a condition of, for example, pit or no pit, positive or negative magnetization, etc., is stored. During read-out, the receiver, under the control of the retrieved clock, properly phased with respect to the incoming data, samples the retrieved signal at the middle of each time slot.

The recording code must be capable to regenerate the bit clock in the player from the read-out signal. To permit this, the number of pit edges per second must be sufficiently large, and in particular the maximum runlength must be as small as possible. Another requirement relates to the 'low-frequency content' of the read signal. This has to be as small as possible. There are two reasons for this. In the first place, the servo systems for track following and focusing are controlled by low-frequency signals, so that low-frequency components of the information signal could interfere with the servo-systems. The second reason is illustrated in figure 1.1, in which the read signal is shown for a clean disc (a) and for a disc that has been soiled, e.g. by fingermarks (b).

This causes the amplitude and average level of the signal to fall. The fall in level causes a completely wrong read-out if the signal falls below the decision level. Errors of this type are avoided by eliminating the low-frequency components with a filter (c), but the use of such a filter is only permissible provided the information signal itself contains no low-frequency components. The high-pass filter should be chosen to pass the signal and to reject the low-frequency noise. It cannot simultaneously do both completely; the filter is chosen as a good compromise between these conflicting



Because of soiling, both amplitude and the signal level decrease; the decision errors that this would cause are eliminated by the filter

Figure 1.1: Read-out signal for six pit edges on the disc

goals. In the Compact Disc system the frequency range from 20 kHz to 1.5 MHz is used for information transmission; the servo systems operate on signals in the range 0 to 20 kHz. [1]

### 1.3 Inter Symbol Interference

When a single pulse is transmitted over a bandwidth-limited system, it is smeared out in time due to convolution with the channel's impulse response. A sample at the centre of a symbol interval is a weighted sum of amplitudes of pulses in several adjacent intervals. This phenomenon is called intersymbol interference (ISI). If the product of symbol time interval and system bandwidth is reduced, the ISI will become more severe. A point may eventually be reached at which this effect becomes so severe that the receiver, even in the absence of noise, can no longer distinguish between the symbol value and the ISI and will start to make errors.[2]

According to Nyquist's criteria for distortionless transmission, all ISI can be eliminated prior to detection by means of a linear filter which equalizes the characteristics of the channel. In practice there are difficulties in providing correct equalization at all times.



## 1.4 Physical Attributes

Tape surface asperities and substrate irregularities may cause variations or fluctuations in the intimacy of head contact, which changes the response at high frequencies much more than at low frequencies. In disc drives, the varying radii of tracks results in a linear density variation of about two to one, and the thickness of the air film, which is a function of the disc-head speed, is also subject to change. In the compact disc, spindle wobble and disc warp mean that the focal plane is continuously moving and the spot quality will vary due to the finite bandwidth of the focus servo. Optimum equalization is difficult to maintain under dynamic conditions, although an adaptive equalizer can be used to follow the dynamic changes. A recording code must be designed to show a certain acceptable ruggedness against the aforementioned dynamic changes of the channel's characteristics.

The physical parameters mentioned above are associated purely with the one-dimensional communication channel approach. Designing for high linear density cannot be the sole objective. In a system approach, both dimensions, track and linear density, should be considered. In magnetic recorders, signal amplitude and signal-to-noise ratio are proportional to the track width. In this sense, linear and track density are traded against each other. For mechanical design simplicity, high-performance digital tape recorders such as the DAT (Digital Audio Tape) recorder are designed to operate at relatively high linear densities but low track density. With

the use of wide tracks with high linear densities, ISI rather than (additive) noise is frequently the limiting factor. This statement is specially true for optical recording systems where noise is virtually absent.

The choice of a particular code depends on numerous factors such as available signal-to-noise ratio, clocking accuracy, non-linearities, and intersymbol interference. Other constraints, such as equipment limitations, ease of encoding and decoding, and the desire to preserve a particular mapping between the source and the code symbols all govern the encoding chosen.

## 1.5 Sofic Systems

Channel constrained codes can be considered subsets of output sequences of so-called sofic systems. A finite directed graph whose edges are labeled by symbols from a finite alphabet defines a sofic system. The sofic system is the set of all bi-infinite sequences of symbols generated by bi-infinite paths in the graph. The graph together with its edge labeling presents a sofic system.

Two special types of sofic systems arise in theory and practice. The easiest to deal with are the shifts of finite type (SFT's). SFT can be presented by a labeled graph in such a way that each element is the label of a unique bi-infinite path in the graph. A practically important class of SFT's are the  $(d,k)$  runlength-limited systems.

The second special type of sofic system is almost finite type (AFT) sofic systems. An AFT sofic system is one that can be presented by a labeled graph in such a way that for all bi-infinite paths in the graph that are labeled are totally separated. A practically important class of AFT systems that are not SFT's are the charge-constrained systems [3]. The charge constrained systems are also known as bounded running sum systems.

## 1.6 Runlength-limited Codes

Recording codes that are based on runlength-limited (RLL) ones are usually used in optical and magnetic disk recording practice. Runlength is the length of time between consecutive transitions and is usually expressed in the number of binary symbols. Runlength-limited sequences are characterized by two parameters,  $(d + 1)$  and  $(k + 1)$ . Relationship between code parameters and the physics of recording systems are summarized for instance in [4].

The parameter  $d$  controls the highest transition frequency and thus has a bearing on intersymbol interference when the sequence is transmitted over a bandwidth-limited channel. In the transmission of binary data it is generally desirable that the received signal is self-synchronizing or self-clocking. Timing is commonly recovered with a phase-locked loop which adjusts the phase of the detection instant according

to observed transitions of the received waveform.

The maximum runlength parameter  $k$  ensures adequate frequency of transitions for synchronization of the read clock. The grounds on which  $d$  and  $k$  values are chosen, in turn, depend on various factors such as the channel response, the desired data rate (or information density), and the jitter and noise characteristics.

Recording codes that are based on runlength-limited sequences have found almost universal application in optical and magnetic disc recording practice. Archetypes are the rate  $1/2$ , ( $d = 2, k = 7$ ) code applied in the IBM3380 rigid disc drive [5], and the EFM code ( $rate = 8/17, d = 2, k = 10$ ) which is employed in the Compact Disc (see Chapter 2 [1]). Runlength-limited codes, in their general form, were pioneered by Berkoff [6], Freiman and Wyner [7], Kautz [8], Gabor [9], Tang and Bahl [10], and notably Franaszek [11].

## 1.7 dk-limited binary codes

A  $dk$ -limited binary sequence, in short, ( $dk$ ) sequence, satisfies simultaneously the following two conditions:

1.  $d$  constraint: two logical 'ones' are separated by a run of consecutive 'zeros' of length at least  $d$ .
2.  $k$  constraint: any run of consecutive 'zeros' is of length at most  $k$ .

If only proviso (1.) is satisfied, the sequence is said to be  $d$ -limited (with  $k = \infty$ ), and will be termed  $(d)$  sequence.

In general, a  $(dk)$  sequence is not employed in optical or magnetic recording without a simple coding step. A  $(dk)$  sequence is converted to a runlength-limited channel sequence in the following way. Let the channel signals be represented by a bipolar sequence  $y_i, y_i \in \{-1, 1\}$ . The channel signals represent the positive or negative magnetization of the recording medium, or pits or lands when dealing with optical recording. The logical 'ones' in the  $(dk)$  sequence indicate the positions of a transition  $1 \rightarrow -1$  or  $-1 \rightarrow 1$  of the corresponding runlength-limited sequence.

The mapping of the waveform by this coding step is known as precoding as shown in table 1.1.

Table 1.1: Precoding

dk-sequence	0	1	0	0	0	1	0	0	1	0	0	0	1	1	0	1
runlength-limited	1	-1	-1	-1	-1	1	1	1	-1	-1	-1	-1	1	-1	-1	1

Sequences that are assumed to be recorded with such a change-of-state encoding step are said to be given in non-return-to-zero-inverse NRZI notation. Coding techniques using the NRZI format are generally accepted in digital optical and magnetic recording practice. The NRZI format is convenient in magnetic recording since differentiation occurs as a part of the physical process in the heads. The original signal is restored in a quite natural fashion by observing the peaks in the retrieved signal. The peaks coincide with the 'ones' of the stored sequence in NRZI notation. The use of the NRZI format in non-differentiating channels, such as the Compact Disc, is less obvious. The RLL sequence has the virtue that at least  $d+1$  and at most  $k+1$  consecutive like symbols occur.

In the table 1.2 some parameters of runlength-limited codes that have found practical application are given.

Table 1.2: Various codes with runlength parameters  $d$  and  $k$ 

$d$	$k$	$R$	Code
0	1	$1/2$	FM, Bi-phase
1	3	$1/2$	MFMM, Miller
2	7	$1/2$	(2,7)
1	7	$2/3$	(1,7)
2	11	$1/2$	3PM
2	10	$8/17$	EFM



For recording codes that map  $m$  information bits onto  $n$  code bits, the minimum separation between transitions is sometimes called the density ratio. The width of the detection window is the duration during which the presence or absence of a transition has to be detected. This detection window is of width  $m/n$  data bit intervals and a large detection window is preferred due to possible bit synchronization imperfections during the read process.

Immink [1] is an excellent book in introducing coding techniques for digital recorders. Siegel and Wolf [12] discuss many types of modulation codes used in magnetic recording and illustrate many examples of application of partial response equalization, sampling detection, and digital signal processing in modulation and coding techniques. Immink [13] is a tutorial to runlength-limited sequences and Marcus [14] provides modulation code design techniques based upon the state splitting algorithm. Pohlmann[15] gives practical details of digital audio and optical disc technology.

## Chapter 2

# LITERATURE SURVEY

### 2.1 Runlength-Limited Error Control Codes

Runlength-limited codes with error correcting capabilities (RLLECC) are better than concatenation of runlength-limited and error control codes. In concatenated coding the constraints are met with an inner constrained code and the error correction is performed by an outer code. RLLECC have reduced complexity and in many cases also improved rate [16]. It has been shown [17], [18] that combined codes can offer much improved efficiency in addition to simpler overall system.

Popplewell and O'Reilly [16] present a new class of RLL codes with a minimum distance 4. The codes are formed by taking an appropriate coset of a linear transparent error correcting code (LTECC) and are relatively simple to implement. [19] solves the problem of selecting a coset which gives the lowest bound on runlength,

i.e., an optimum coset.

Ferreira and Lin [20] emphasized systematic analytical code construction procedures that work for any  $d$  constraint and implement easily coder and decoder. The block codes are capable of detecting and correcting single bit-errors, single-peak shift-errors, double adjacent-errors and multiple adjacent erasures.

Helberg and Ferreira [21] develop five new combined codes. Three methods are described for introducing error correction capabilities. In the first one some distance building characteristics are incorporated into the constrained code. In this way the concatenability of the constrained sequences is taken care of by the original finite state transition diagram that describes the constraints. In the second method the constrained symbols are mapped onto a shift register graph to enhance the length before emergence of the distance building paths. In the second method care must be taken to ensure that the constraints are preserved. Hamming distance preserving mapping is the third method. Codes are generated by shift registers and nonlinear combinatorial logic and empirical techniques.

Helberg and Ferreira [21] develop a new  $(d,k) = (1,5)$  code that has the fewest number of states and has a higher rate and density ratio than known codes. The known codes have better error correcting capabilities. To achieve higher error correcting capabilities, either complexity, rate, density ratio or a combination of these parameters have to be sacrificed. A new  $(d,k)=(1,3)$  convolutional code increases the free distance by increasing the number of states from three to four. This increase

in the free distance has advantages when using soft decision Viterbi decoding.

Ferreira, Wright and Nel [22] discuss the concept of Hamming-distance preserving RLL codes. Fredrickson and Wolf [23] concentrate on RLL codes that can detect shifted 'ones' in the  $(d,k)$ -constrained bit stream. Lee and Wolf [24] present a general algorithm for constructing a single error-correcting RLL code. The Euclidean distance between RLL sequences in the physical context of the noisy magnetic recorder is investigated by Immink [25].

Patapoutian and Kumar[26] construct  $(d,k)$  error-correcting block codes by employing linear block codes. The scheme allows asymptotically reliable transmission. For the same error-correction capability, the rate is comparable to the parent linear block code. The single-error correcting code is asymptotically optimal.

The errors obtained in magnetic recording channels are the following:

1. 0 is changed into a 1
2. 1 is changed into a 0
3. 1 is shifted one position in either direction

Ytrehus [27] provides the construction of constrained codes for the mixed error channel. Mixed error channel combines the binary symmetric channel and the peak shift channel see table 2.1 for different channel models. The codes achieve code rates close to the  $(d,k)$  capacity.

Table 2.1: Set of Error Types in Different Channel Models

Channel	Set of error types
Binary Symmetric Channel (BSC)	$0 \rightarrow 1, 1 \rightarrow 0$
Asymmetric Channel (ASC)	$1 \rightarrow 0$
Peak Shift Channel (SC)	LeftShift, RightShift

Bassalygo [28] considers bounds on the size of a maximal such code. A lower bound in the case of  $t$ -error correction is derived by considering the coset of a  $t$ -error correcting BCH (Bose-Chaudhuri Hocquenghem) code of length  $n$  that contains the maximum number of constrained  $n$ -tuples.

Ytrehus [29] gives upper bounds derived on the number of codewords in error correcting  $(d,k)$  constrained block codes. The results suggest that in such codes fairly long block lengths are necessary to achieve code rates close to capacity.

Immink [30] describes a new technique for constructing fixed length  $(d,k)$  runlength-limited block codes. The basic idea of the new construction is to uniquely represent each source word by a  $(d,k)$  sequence with specific predefined properties, and to construct a bridge of  $\beta$ ,  $1 \leq \beta \leq d$ , merging bits between every pair of adjacent words. An essential element of the new coding principle is look ahead. The new constructions have the virtue that only one look-up table is required for encoding and decoding.

## 2.2 DC-free Sequences (Bounded Running Sequences)

According to Immink [1] binary sequences with spectral nulls at zero frequency have wide application in optical and magnetic recording systems. DC balanced codes can be used to reduce interaction between the data written on the disc and the servo

systems that follow the track. The low frequency disturbances for example due to finger prints may cause completely wrong read-out if the signal falls below the decision level. Errors of this type are avoided by high pass filtering. In balanced codes each symbol has the same number of 0's as 1's. Balanced codes are assumed to have null at dc. [31] presents a thorough treatment of balanced codes.

Practical coding schemes devised to achieve suppression of low-frequency components are by mostly block codes. Basically three approaches have been used for dc-balanced codes which are zero-disparity, low-disparity and polarity-disparity. The disparity is defined as excess number of 'ones' over the number of zero's in the codeword. Zero-disparity codewords contain equal number of 'ones' and 'zeros'. In low-disparity the translations are not one-to-one. In polarity-disparity for codeword of  $n$  digits the  $(n-1)$  source symbols are supplemented by one symbol called the polarity bit. The encoder has the option to transmit the  $n$ -bit words without modification or to invert all symbols [1].

### 2.3 Running Digital Sum (RDS) Constrained

A code is said to be Running Digital Sum (RDS) constrained if all possible information sequences can be encoded into sequences such that the sum is bounded by some given positive integer. Pierobon [32] shows that the power density function of an encoded sequence vanishes at zero frequency if, and only if, the encoder is a finite

running digital sum encoder. Justesen [33] and Immink [34] show that sum variance could be a valuable criterion of the low-frequency characteristics of a channel code and sum variance of a dc-free sequence is simple to calculate.

Blaum et al [35] present a new approach for encoding any string of information bits into a sequence having bounded running digital sum. The results are better than the previously known values of the RDS for same rate. These codes also provide error correcting capabilities. Cohen and Litsyn [36] give the asymptotic bounds on parameters of error-correcting codes with small running digital sum. The bounds demonstrates the existence of long codes with good error-correcting properties.

Deng et al [37] present a class of convolutional codes which have both DC-free and error correcting properties and which can be encoded and decoded easily.

Barg [38] states that the codes formed by vectors of values of characters of polynomials with argument running over (a subset of) points of a finite field are a frequent object of investigation, since one can estimate their parameters via the classical inequalities for the values of exponential sums. In [39] it is observed that applying estimates of incomplete sums, one can prove that some codes of this type have small running digital sum compared to the block length. Barg [38] derives an extension of the results of [39] to nonprime extension fields and to the nonbinary case, which leads to new families of error-correcting dc-constrained codes.



## 2.4 Problem Definition

Conventional linear Error Correcting Codes do not possess desirable recording code properties since they have unbounded running disparity and have unlimited runlengths which can lead to several problems. To combat noise present in the recording channel, several schemes for error-correction have previously been considered employing error-correcting code surrounding an inner modulation code.

Runlength constraints can be used for error correction and DC-free sequences. Generally, codes which combine error correction and runlength constraints are better than concatenated schemes. Runlength constraints can be applied on the running digital sum or on the maximum/minimum consecutive sequence of 1's or 0's. Codes with  $d_{min} = 4$  are used in runlength limited sequences. Channels that can accept only a subset of all possible input sequences, called constrained channels, would be considered here.

In transparent codes the logical complement of the code is also a code vector. Transparent codes are not discussed in the literature in sufficient detail. Several properties of transparent codes are investigated. Reflective codes are subclass of transparent codes. Several properties of reflective codes are also investigated. Study of transparent codes also helps in the construction of runlength limited codes. Several runlength codes which are transparent in nature have been developed and their properties such as maximum runlength, length of code vector, length of information

vector, weight enumeration, duality are discussed.

In the literature there is a detailed study of dual codes and their properties. But there seems to be a lack of study of transparent codes. It is envisaged to investigate transparent codes and their properties in particular with respect to weight distribution. Popplewell [16] defines transparent codes which are of  $d_{min} = 4$  and satisfy certain conditions. We start out to investigate a subclass of transparent codes defined by Popplewell with the aim of improving the results by Popplewell et al.

Linear transparent block codes can be modified by adding to each codeword a modification vector, so that runlength constraints are satisfied. The new code can be decoded quite easily and it shares the transparent property with the parent code as well. The powerful theory of binary linear block codes can be applied to the modified codes.

## Chapter 3

# GENERAL PROPERTIES OF TRANSPARENT CODES

In this chapter linear codes and Hamming weight are briefly explained. The concept of logical inverse is discussed in detail. Generator Matrix  $G$  and parity check matrix  $H$  are discussed with some new definitions and propositions. Reflective codes are also briefly discussed in this chapter. Finally runlength is discussed in much detail.

### 3.1 Galois Field

Finite fields do not exist for any arbitrary number of elements. In general, they exist only when the number of elements is a prime number or is a power of a prime

number. The former are called prime fields, while the latter are called extension fields over the prime field.

**Definition 3.1.1** *Let  $R$  be a set,  $+$  and  $\cdot$  be two binary operations on  $R$  and  $0, 1$  be special elements of  $R$ . If further that :*

1.  $(R, +, 0)$  is an abelian group
2.  $(R, \cdot, 1)$  is an abelian group
3. the distributive laws hold: For every  $a, b, c \in R$

$$a(b + c) = ab + ac$$

$$(a + b)c = ac + bc$$

then  $(R, +, \cdot, 0, 1)$  is called a field. If  $R$  is finite and has  $q$  elements then the field is called finite or a Galois field and is denoted by  $GF(q)$ .

## 3.2 Linear Codes

**Definition 3.2.1** *A linear code of length  $n$  is a subspace of the vector space  $GF(q)^n$  where  $GF(q)$  is some Galois field.*

In other words, a linear code is a nonempty set of  $n$ -tuples over  $GF(q)$  called codewords such that the sum of two codewords is again a codeword, and the product of any codeword by a field element is a codeword. For any linear code, the all-zero

word, as the vector-space origin, is always a codeword. More directly, if  $c$  is a codeword then its additive inverse  $-c$  is a codeword too and hence  $c + (-c) = 0$  is a codeword.

**Definition 3.2.2** *Let  $K$  be a subset of  $GF(q)^n$ . The Hamming weight  $w(a)$  of any word of  $GF(q)^n$  is equal to the number of nonzero components in  $C$ . The minimum weight  $w(K)$  is the smallest weight of any nonzero word of  $K$ .*

*The Hamming distance between two words  $a$  and  $b$  is the number of places where they differ, and is denoted by  $d(a, b)$ .*

*The minimum distance  $d(K)$  of  $K$  is the smallest Hamming distance of two distinct words of  $K$ ; i.e.,*

$$d(K) = \min\{d(a, b) \mid a, b \in K, a \neq b\} \quad (3.1)$$

**Example 3.2.1** *The Hamming distance of two pairs of words defined over  $GF(2)$  and  $GF(3)$  respectively is*

$$d(10111, 00101) = 2 \text{ and } d(0122, 1220) = 3.$$

If  $\mathbf{u} = u_1 \dots u_n$ ,  $\mathbf{v} = v_1 \dots v_n$  are vectors (with components from a field  $F$ ), their scalar product is

$$\mathbf{u} \cdot \mathbf{v} = u_1 v_1 + \dots + u_n v_n \quad (3.2)$$

evaluated in  $F$ . If  $\mathbf{u} \cdot \mathbf{v} = 0$ ,  $\mathbf{u}$  and  $\mathbf{v}$  are called orthogonal.

**Definition 3.2.3** If  $C$  is an  $(n,k)$  linear code over field  $F$ , its dual or orthogonal code  $C^\perp$  is the set of vectors which are orthogonal to all codewords of  $C$ :

$$C^\perp = \{\mathbf{u} \mid \mathbf{u} \cdot \mathbf{v} = 0 \text{ for all } \mathbf{v} \in C\} \quad (3.3)$$

**Proposition 3.2.1** Let  $C$  be a linear code. Then every coset corresponding to  $C$  has the same minimum distance as  $C$ .

**Proof:** Assume that  $C$  is a linear code and  $C'$  is a coset obtained by adding a non-codeword  $m$  to every codeword in  $C$ . Suppose  $a' \in C'$ , then  $a' = a + m$  for some  $a \in C$ . The distance  $d(a, 0)$  would change to  $d(a + m, m) = d(a', m)$ . Since  $d(a, b) = d(a + b, 0)$  it follows that  $d(a + m, m) = d(a + m + m, 0) = d(a, 0)$ . Thus even after addition of vector  $m$  the minimum distance of coset  $C'$  is same as of the original code.  $\square$

### 3.3 Quasi-Cyclic Codes

**Definition 3.3.1** A Code  $C$  is quasi-cyclic (QC) if it is linear and if a right circular shift of  $p$  positions of any codeword of  $C$  is also in  $C$ . i.e., whenever  $(c_0, c_1, \dots, c_{n-1})$  is in  $C$  then so is

$$(c_{n-p}, c_{n-p+1}, \dots, c_0, c_1, \dots, c_{n-p-2}, c_{n-p-1}).$$

If  $p = 1$ ,  $C$  is called a cyclic code.

**Example 3.3.1** For  $p=2$ , a 2-cyclic code, with  $n = 6$ ,  $k = 3$  is defined by the generator matrix

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

All the codewords of the (6,3) 2-cyclic code are shown in table 3.1.

Table 3.1: (6,3) 2-cyclic code

<i>Message</i>	<i>Codeword</i>
0	000000
1	000011
2	001100
3	001111
4	110000
5	110011
6	111100
7	111111



## 3.4 Transparent Codes

**Definition 3.4.1** *A code  $C$  over  $GF(2^m)$  is a transparent or self-complementary code if the logical inverse of every codeword is also a codeword.*

### 3.4.1 Logical Inverse

The logical inverse of the codeword over a binary field is the one's complement. To find the logical inverse of a binary extension field element, find its binary representation, find its one's complement and find the corresponding symbol over the extension field.

**Example 3.4.1** *If the codeword of length  $n = 8$  over  $GF(2)$  is given by 11001100, then the logical inverse would be 00110011.*

**Example 3.4.2** *Assume now that a codeword of length  $n$  is given in exponential representation:*

$$w = 0\alpha^0\alpha^1\alpha^3\alpha^2\alpha^6\alpha^4\alpha^5$$

*This codeword can be described in binary representation using the table (3.2).*

Table 3.2: Symbols of  $GF(2^3)$  expressed in binary form

<i>Message</i>	<i>Codeword</i>	<i>Logic Inverse</i>
0	000	$\alpha^5$
$\alpha^0$	001	$\alpha^4$
$\alpha^1$	010	$\alpha^6$
$\alpha^3$	011	$\alpha^2$
$\alpha^2$	100	$\alpha^3$
$\alpha^6$	101	$\alpha^1$
$\alpha^4$	110	$\alpha^0$
$\alpha^5$	111	0

The codeword  $w$  can be written in terms of binary numbers as:

$$w = (000)(001)(010)(011)(100)(101)(110)(111)$$

The codeword  $w'$ , the complement of the codeword  $w$ , in binary representation would be:

$$w' = (111)(110)(101)(100)(011)(010)(001)(000)$$

Finally finding the corresponding symbols in exponential representation:

$$w' = \alpha^5 \alpha^4 \alpha^6 \alpha^2 \alpha^3 \alpha^1 \alpha^0 0$$

To find the complement of any code  $C$  over a binary extension field,  $C$  is expressed in terms of binary symbols. Then the logical complement is obtained by changing zero's to one's and one's to zeros.

Note that the complement of a field element must always be a different field element; i.e., the notion of a self-complemented codeword does not exist. This means that any field, each of which has a complement, must have an even number of elements. This excludes all finite fields, except for the binary field and its extension fields.

**Proposition 3.4.1** *A linear code over  $GF(2)$  is transparent iff it contains the all-one codeword.*

**Proof:** Let  $C$  be a transparent code. Since  $C$  is a linear code it contains the all-zero codeword. The logical inverse of the all-zero word is the all-one word. Hence

the all-one is always a codeword of a linear transparent code. Conversely, let  $C$  be a linear code containing the all-one word codeword. If this codeword is added to any codeword, the complement of the codeword is obtained and since by linearity the sum of any two codewords is also a codeword it follows that the complement is included. Hence a linear code  $C$  containing the all-one word is a transparent code.

□

We make the basic assumption that from now on all transparent codes considered are linear codes unless otherwise stated.

**Example 3.4.3** *If  $c = 11001100$ . is a codeword of some code of length  $n$  over  $GF(2)$  the complement is obtained by adding the all-one code vector:*

$$11001100 + 11111111 = 00110011$$

**Example 3.4.4** *Let  $c = \alpha^3\alpha^10\alpha^0$ . be a codeword of some code of length  $n$  over  $GF(2^3)$ . The complement is obtained by adding the all-one code vector  $\alpha^5\alpha^5\alpha^5\alpha^5$*

$$c = \alpha^3\alpha^10\alpha^0$$

$$\alpha^3\alpha^10\alpha^0 + \alpha^5\alpha^5\alpha^5\alpha^5 = \alpha^2\alpha^6\alpha^5\alpha^4$$

*Thus the complement of  $c$  is given by  $c' = \alpha^2\alpha^6\alpha^5\alpha^4$ .*

**Proposition 3.4.2** *A code over  $GF(2)$  is transparent iff every column of the generator matrix  $G$  contains an odd number of 1's.*

**Proof:** A transparent code always contain the all-one codeword. To get the all-one codeword, the sum of every column must be one. Thus every column of the generator matrix of a transparent code must contain an odd number of ones. Conversely, if every column of the generator matrix contains an odd number of 1's, then the addition of all rows of  $G$  yield the all-one codeword. Since the all-one codeword is present, the code is transparent.  $\square$

**Proposition 3.4.3** *A code over  $GF(2)$  is transparent if every row of a parity-check matrix  $H$  has an even number of 1's.*

**Proof:** If  $c$  is a codeword then  $cH^t$  must be the all zero vector, i.e., the  $c$  product  $cH^t$  must be zero. This will clearly be true if the row has an even number of 1's since the mod 2 sum of an even number of 1's is always zero.  $\square$

For instance, if  $n=7$  then  $c = 1111111$ , then  $[1111111]H^t$  could only be zero if every row of  $H$  has an even number of 1's.

**Proposition 3.4.4** *The dual code of a transparent code is not transparent*

**Proof:** It will be proved that Hamming codes are transparent codes but their duals, the simplex codes, are not. Thus the dual of a transparent code is not always transparent.  $\square$

## 3.5 Reflective Codes

**Definition 3.5.1** *A code  $C$  is reflective, if  $C$  and its dual  $C^\perp$  are both transparent.*

The subclass of transparent codes proposed by Popplewell and O'Reilly belongs to the class of *reflective codes*. Hamming codes are not reflective because the dual code of a Hamming code is not transparent.

**Proposition 3.5.1** *The generator matrix  $G$  and parity-check matrix  $H$  of a reflective code have the following properties:*

1. *every row of  $G$  and  $H$  has an even number of 1's.*
2. *every column of  $G$  and  $H$  has an odd number of 1's.*

**Proof:** By Proposition (3.4.2) the  $G$  matrix of a transparent code has an odd number of 1's and by proposition (3.4.3)  $H$  has even number of 1's in every row. Since the dual of reflective code is also transparent. It follows that the  $G$  matrix of a reflective code has an even number of 1's in every row and the  $H$  matrix has an odd number of 1's in every column.  $\square$

## 3.6 Runlength

Runlength constraints are very important in storage devices. If the runlengths are very short then many distortions are caused by inter-symbol interference and if the runlengths are long enough then problems of synchronization and low frequency components are encountered. Low frequency components must be avoided because a lower spectrum is occupied by the control signals.

**Definition 3.6.1** A (possibly infinite) sequence  $s$  over the (finite) alphabet  $A$  has runlength  $RL(s, a)$  if it is the maximum number of consecutive symbols  $a \in A$  in  $s$ .

The sequence  $s$  has maximal runlength  $RL_{max}(s)$  if

$$RL_{max}(s) = \max_{a \in A} \{RL(s, a)\} \quad (3.4)$$

and unlimited runlength if  $RL(s, a)$  is the length of the sequence.

If  $s$  is a sequence of words of  $C \in A^n$ , i.e.,  $s \in (A^n)^*$ , where  $(A^n)^*$  is the set of all sequences of words of  $C$ , then

$$RL(C, a) = \max_{s \in (A^n)^*} \{RL(s, a)\} \quad (3.5)$$

and

$$RL_{max}(C) = \max_{a \in A} \{RL(C, a)\} \quad (3.6)$$

The maximal runlength of a sequence of codewords can be determined from the  $G$  matrix of the given code. To this end we use the following proposition.

**Proposition 3.6.1** A code  $C$  contains a nonzero codeword of Hamming weight  $w$  or less iff a linearly dependent set of  $w$  columns of  $H$  exist.

**Proof:** For any codeword  $c$ ,  $cH^t = 0$ . Let  $c$  have weight  $w$ . Drop the components of  $c$  that are zero. This is a linear-dependence relation in  $w$  columns of  $H$ . Hence,  $H$  has a linearly dependent set of  $w$  columns. Conversely, if  $H$  has a linearly dependent set of  $w$  columns, then a linear combination of at most  $w$  columns is equal to zero.

These  $w$  non-zero coefficients define a vector of weight  $w$  or less for which  $cH^t = 0$ .

□

**Definition 3.6.2** *Let  $G$  be a (fixed) generator matrix of a binary linear transparent code. Then*

- *RS is the maximum number of consecutive columns of  $G$  including the first column (Start).*
- *RE is the maximum number of consecutive columns of  $G$  including the last one (End).*
- *RM is the maximum number of consecutive columns of  $G$  excluding the first and last one (Middle).*

Now let  $\xi$  be the set of all permutations of columns of  $G$ . Then

$$RS_{min}(C) = MIN\{RS|G \in \xi\}$$

$$RE_{min}(C) = MIN\{RE|G \in \xi\}$$

$$RM_{min}(C) = MIN\{RM|G \in \xi\}$$

RS

**Proposition 3.6.2** *A subset  $C \in A^n$  has maximal runlength*

$$RL_{max}(C) = MAX\{RE + RS, RM\} \quad (3.7)$$



where  $RS(a)$ ,  $RE_{min}(a)$  and  $RM(a)$  are the maximum numbers of consecutive symbols  $a$  in any word of  $C$  at the start, end and anywhere in the middle, respectively.

If  $RS(a) = RE(a) = RM(a) = n$  then  $C$  has unlimited runlength.

**Proof:** If a sequence of codewords does not contain any all-one or all-zero codewords then maximum runlength could be expressed as the maximum number of consecutive symbols (1/0's) in middle of the codeword or it could be the sum of maximum consecutive symbols in the end of preceding codeword and the start of succeeding codeword.  $\square$

**Proposition 3.6.3** *If  $C$  is a  $(n, k)$  linear block code then  $RL_{max} \leq 2k$ .*

**Proof:** Note from matrix theory that since the rank of an  $(n, k)$  matrix corresponds to the number of linearly independent rows or columns, the rank  $\leq MIN\{n, k\} = k$  for all error control codes, whence  $RS_{min} \leq k$ ,  $RM_{min} \leq k$  and  $RE_{min} \leq k$ . Since  $RL_{max} = MAX\{RS + RE, RM\}$ , it implies that  $RL_{max} \leq 2k$ .  $\square$

Linear transparent codes over  $GF(2)$  have unlimited runlength due to the inclusion of the all-zero and the all-one codeword. Therefore to produce ECCs having limited runlengths we need to find a set of codewords which does not contain the all-zero or all-one word whilst at the same time will retain the minimum distance properties of the parent ECC.

## Chapter 4

# TRANSPARENT CODES

Some of the popular codes are found to be transparent for instance Hamming codes, Repetition code, Perfect codes, Golay code. This chapter provides a brief overview of these codes and some propositions regarding properties of transparent codes.

### 4.1 Hamming Codes

**Definition 4.1.1** *A binary Hamming code is defined by its code length  $n$  and message length  $k$ ,  $n = 2^m - 1$  and  $k = 2^m - m - 1$  where  $m \geq 2$ . The dual of a Hamming code is called simplex code or maximum length code. See Fig.(4.1).*

Length:	$n = 2^m - 1 = n^\perp$
Information symbols:	$k = 2^m - m - 1 = m^\perp$
Minimum distance:	$d = 3$
Length of information message for the dual Hamming code:	$m = k^\perp$
Error control capability:	Perfect codes for correcting single errors

Figure 4.1: Properties of Hamming Codes

**Proposition 4.1.1** *A parity check matrix  $H$  of a binary  $(n,k)$  Hamming code has every column different from any other one and none is the zero column vector. Since  $n = 2^m - 1$  all nonzero vectors of length  $m$  are present.*

**Proof:** This is simple proof see [40]. □

**Proposition 4.1.2** *Every row of parity check matrix  $H$  of a binary Hamming code  $(n,k)$  has weight of  $2^{m-1}$ .*

**Proof:** All the non-zero vectors of length  $m$  are present in the columns of the  $H$  matrix. This implies that half the entries of every row of  $H$  are 1. Since the total number of columns are  $2^m$  (including all-0), it follows that the weight of every row is half of  $2^m$ ; i.e., the weight of every row of the  $H$  matrix is  $2^{m-1}$ . □

For example see fig. 4.2 and fig. 4.3.

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

Figure 4.2: Parity-check matrix of the Hamming code (7,4)

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

Figure 4.3: Parity-check matrix of the Hamming code (15,11)

**Proposition 4.1.3** *Any binary  $(n,k)$  Hamming code  $C$  is transparent.*

**Proof:** For Hamming code  $(n, k)$  where  $n = 2^m - 1$  and  $k = 2^m - m - 1$  all rows of the parity-check matrix have weight  $2^{m-1}$  by Proposition (4.1.2). But  $2^{m-1}$  is always even by Proposition (4.1.2). This implies that all rows of the parity-check matrix of a Hamming code have an even number of 1s. Hence by Proposition (3.4.3) the all-1 codeword exists. Thus Hamming codes are transparent.  $\square$

**Proposition 4.1.4** *The simplex  $(n,k)$  code (dual of Hamming code) is not transparent.*

**Proof:** The parity-check matrix of the dual of a Hamming code is the generator matrix of the Hamming code. Since all non-zero vectors of length  $m$  are present in the columns of the parity-check matrix all non-zero vectors of length  $m$  are present in the columns of generator matrix of the simplex code. Since all the non-zero vectors of length  $m$  are present it follows that some columns of the generator matrix have even number of one's. Since even number of one's are present in the columns of generator matrix, it shows that this code could not be the transparent code. Thus Simplex code is not transparent. See fig. 4.4 and fig.4.5.  $\square$

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

Figure 4.4: Parity-check matrix of the dual Hamming code (7,4)



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

Figure 4.5: Parity-check matrix of the dual Hamming code (15,11)

## 4.2 Repetition Codes

**Definition 4.2.1** *The repetition code  $(n,1)$  contains two codewords, the sequence of  $n$  zeros and the sequence of  $n$  ones.*

**Proposition 4.2.1** *Every repetition code is a transparent code.*

**Proof:** In repetition code, there are two code words which are all 1's and all 0's. These two are 1's complement of each other. Thus the repetition code is transparent.  $\square$

For example the codewords of the  $(5,1)$  repetition code are 00000 and 11111. These are 1's complements of each other. Thus the repetition code is a transparent code.

## 4.3 Perfect Codes

Visualize a small sphere about each of the codewords of a code, each sphere with the same radius (an integer). Allow these spheres to increase in radius by integer amounts until they cannot be made larger without causing some spheres to intersect. That value of the radius is equal to the number of errors that can be corrected by the code. It is called the packing radius of the code. Now allow the radii to continue to increase by integer amounts, until every point in the space is contained in at least one sphere. That radius is called the covering radius of the code. The packing

radius and the covering radius are equal in case of perfect codes.

**Definition 4.3.1** *A perfect code is one for which there are equal radius spheres about the codewords that are disjoint and completely fill the space.*

Examples of perfect codes are the  $(n, 1)$  repetition code when  $n$  is odd and the Hamming Code.

**Proposition 4.3.1** *A perfect  $t$ -error correcting binary  $(n, k)$  code exists, if the numbers  $n$ ,  $k$  and  $t$  satisfy the following equation.*

$$2^n = \left[ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} \cdots \binom{n}{t} \right] 2^k$$

**Proof:** For a  $(n, k)$  linear perfect code over  $GF(2)$ , the number of codewords is  $2^k$ . Codes having all the words in spheres of radius 1 are single error correcting. There could be  $n$  number of words at a distance 1 from any code word. Thus we have a total number of words equal to  $n + 1 = \binom{n}{0} + \binom{n}{1}$  at distance less than or equal to 1 in one sphere. So the total number of words in all spheres of radius 1 becomes  $2^k \times (n + 1)$ . In perfect codes all the words are within spheres. Thus the total number of words  $2^n$  is equal to the words within all spheres  $2^k(n + 1)$ . For spheres of radius 2, there could be  $\binom{n}{0} + \binom{n}{1} + \binom{n}{2}$  number of words in (or on) the sphere. Thus total number of words  $2^n$  becomes  $2^k \left[ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} \right]$ . More generally, spheres of radius  $t$ . contain  $\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t}$  words. Thus the total number of words in all spheres would be  $2^k \times \left( \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{t} \right)$ . Thus  $2^k$  spheres contain all

$2^n$  points of the space.

$$2^n = \left[ \binom{n}{0} + \binom{n}{1} + \binom{n}{2} \cdots \binom{n}{t} \right] 2^k$$

□

Consider now for which parameters a binary perfect code could exist. If  $t = 1$ , or  $d = 3$  we have  $(1 + n)2^k = 2^n$  so that  $n = 2^{n-k} - 1$ . Letting  $m = n - k$ , we have the parameters  $n = 2^m - 1$ ,  $k = 2^m - 1$ , which are satisfied by the Hamming code. So for this family of possible parameters, we have perfect codes with these parameters. If  $t = 2$ , we have  $(1 + n + \binom{n}{2})2^k = 2^n$ , so that  $1 + n + \binom{n}{2} = 2^{n-k}$ . The first  $n$ , for which  $1 + n + \binom{n}{2}$  is a power of 2, is  $n = 5$ . A perfect code with these parameters is the  $(5, 1)$  repetition code. If  $t = 3$ , a sphere contains  $1 + n + \binom{n}{2} + \binom{n}{3}$  points and this is a power of 2 for  $n = 23$ . This yields the a  $(23, 12)$  binary Golay code with  $d = 7$ .  $1 + 2n + 4\binom{n}{2}$  is a power of 3 for  $n = 11$  and the perfect Golay ternary  $(11, 6)$  code with  $d = 5$  is obtained.

**Proposition 4.3.2** *The Golay code  $(23, 12)$  with  $d = 7$  is a transparent code.*

**Proof:** The weight distribution of the  $(23, 12)$  Golay code with  $d = 7$  is shown in table 4.1 [40]. Since the all 1's codeword is present, the  $(23, 12)$  Golay code is transparent. □

Table 4.1: Weight Distribution of Golay Code  $(n,k,d) = (23,12,7)$ 

<i>Weight</i>	<i>#ofequiweight Codewords</i>
0	1
7	253
8	506
11	1288
12	1288
15	506
16	253
23	1

**Proposition 4.3.3** *All binary perfect codes are transparent.*

**Proof:** The repetition code  $(n, 1)$  when  $n$  is odd, and Hamming Codes  $(n, k)$  are perfect and transparent. The Golay Code  $(23, 12)$  is also perfect code. Thus all linear binary perfect codes are transparent.  $\square$

# Chapter 5

## MODIFIED CODES

In this chapter modification of codes is discussed. Algorithm for finding a modification vector over binary fields and over binary extension fields are explained and examples are given. Generator Matrix transformations from binary extension fields to binary fields are developed.

### 5.1 Modification of Linear Codes

It has been proved that the transparent code contains the all-one codeword and that its existence causes unlimited runlength sequences. A special vector called modification vector is added to every codeword to limit the runlength.

The error correction capability depends upon the minimum distance of the code [41], [42]. The error correction capability of the code is also preserved after adding

an  $n$ -bit vector  $m \in \mathcal{C}$  with every codeword. This means that it is possible to encode an information sequence by standard encoding methods but before transmitting the codeword, one adds a particular constant modification vector so that the codeword transmitted along the channel is now runlength limited. At the receiver side the inverse operation is performed; thus, before any error correction is attempted, the received word is first added to  $m$  and then standard error control decoding can be performed. This procedure is illustrated in Figure 5.1.



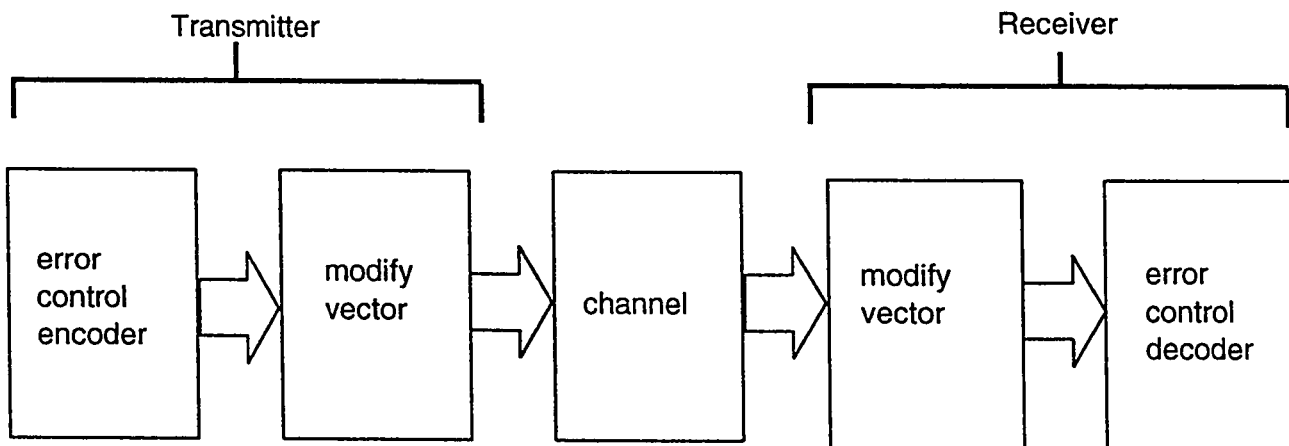


Figure 5.1: Block diagram of a system employing runlength limited ECCs

## 5.2 Modification Vector

A modification vector is needed to satisfy the runlength-limited constraints on the codewords. The error correction capability is conserved due to the principle of linearity. Undesired runlengths are avoided by adding suitable modification vector.

Given a generator matrix  $G$  for the  $ECC$ . First a systematic generator matrix is determined by row operations and column operations. Then the systematic parity-check matrix is obtained from the systematic generator matrix. The parity check matrix  $H$  is obtained from the systematic parity-check matrix. Then row operations are performed on the parity-check matrix such that the desired  $RL_{max}$  is obtained. Some of the rows of  $H$  are needed to satisfy the constraints on  $RS_{min}$ ,  $RE_{min}$  and  $RM_{min}$ . A new matrix  $s$  is obtained which contains the rows of  $H$  which are necessary to satisfy the runlength constraints. Modification vector are obtained by solving a set of equations. There are many possible solutions. A suitable modification vector is chosen depending on the application.

**Algorithm 5.2.1** *Get  $H$  matrix for the given code  $C$ .*

1. *Rearrange  $H$  by performing row operations to produce  $\tilde{H}$  so that it satisfies the following conditions:*

(a)  *$\tilde{H}$  has at least one row  $z$  such that*

$$\tilde{h}_{z,i} = 0 \text{ for all } i = RS_{min} + 2, RS_{min} + 3, \dots, n \quad (5.1)$$

(b)  $\tilde{H}$  has at least one row  $z$  such that,

$$\tilde{h}_{z,i} = 0 \text{ for}$$

$$i = 1, 2, \dots, p, \text{MinRL}_{\max} + p + 2, \text{MinRL}_{\max} + p + 3, \dots, n \quad (5.2)$$

where  $p = 1, 2, \dots, n - \text{MinRL}_{\max} - 2$

(c)  $\tilde{H}$  has at least one row  $z$  such that,

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, \dots, n - RE_{\min} - 1 \quad (5.3)$$

2. If  $\tilde{H}$  does not satisfy all these conditions then there does not exist a coset that satisfies the bound on  $\text{MinRL}_{\max}$  and accordingly one or more of the bounds on  $RE_{\min}$ ,  $RS_{\min}$  or  $RM_{\min}$  must be increased until all the conditions of 5.1, 5.2 and 5.3 are satisfied. The particular bound which is increased depends upon which condition is not satisfied.
3. When  $\tilde{H}$  satisfies all the conditions in (2) then form a new matrix  $S$  which is constructed from all rows of  $\tilde{H}$  that satisfy at least one of the conditions (a), (b) and (c).
4. Solve the matrix equation

$$mS^t = \mathbf{1} \quad (5.4)$$

Any solution of this equation will produce a modification vector  $m$  which satisfies the predetermined bounds on  $\text{MinRL}_{\max}$ .

The parity check matrix may be put in this form by using a method similar to Gaussian elimination; the procedure involves a maximum of

$$(n - k)(n - k - 1) \approx (n - k)^2 \text{ row operations} \quad (5.5)$$

For a given code the complexity of the procedure therefore increases with the square of the number of check bits.

See the flow chart in figure(5.2).

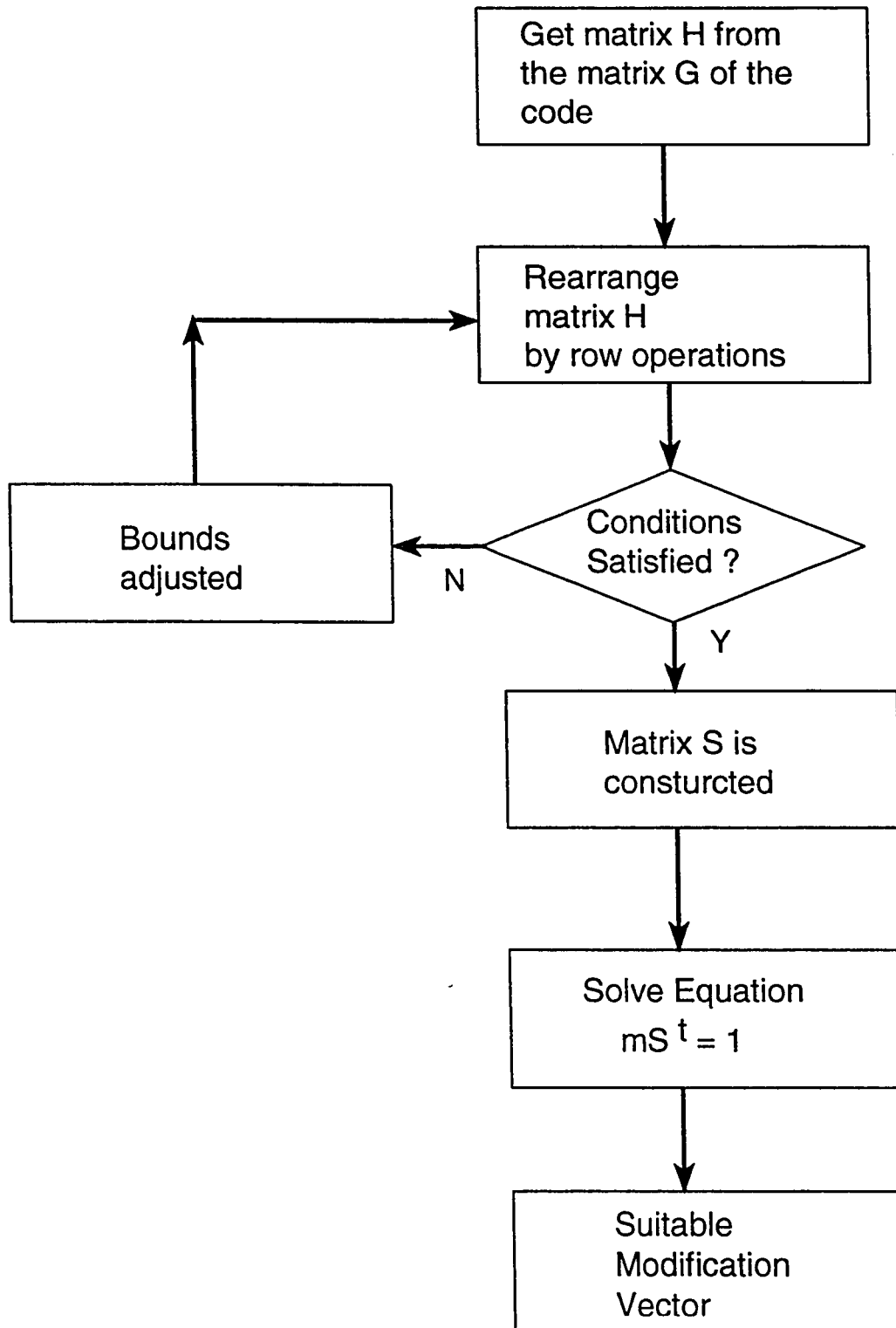


Figure 5.2: Algorithm for suitable modification vector

**Proof: (Outline)** To find a modification vector,  $m$ , which forms a coset satisfying the predetermined runlength bounds we must find  $m$  so that it simultaneously satisfies the following conditions:

$$\begin{aligned}
m_1 \cdots m_{RS_{min}+1} &\neq C_{1,RS_{min}+1}^i \\
m_2 \cdots m_{MinRL_{max}+2} &\neq C_{2,MinRL_{max}+2}^i \\
m_3 \cdots m_{MinRL_{max}+3} &\neq C_{3,MinRL_{max}+3}^i \\
\dots &\dots \dots \\
\dots &\dots \dots \\
\dots &\dots \dots \\
m_{n-MinRL_{max}-2} \cdots m_{n-1} &\neq C_{n-MinRL_{max}-2,n-1}^i \\
m_{n-MinRL_{max}-1} \cdots m_n &\neq C_{n-MinRL_{max}-1,n}^i
\end{aligned} \tag{5.6}$$

for all  $i = 0, 1, \dots, 2^k - 1$ , where  $C_{j,k}^i$  is the set of words consisting of codeword bits  $j$  to  $k$ .

If the received word  $r$  is a member of the set of error control codewords and  $H = [h_{i,j}]$  is the parity check matrix then  $rH^t = 0$ . Conversely if  $r$  is not a member of the set of codewords then  $rH^t \neq 0$ . So, since  $m$  is not a codeword it must satisfy

at least one of the following equations:

$$\begin{aligned}
\sum_{i=1}^n m_i h_{1,i} &= 1 \\
\sum_{i=1}^n m_i h_{2,i} &= 1 \\
&\dots \\
\sum_{i=1}^n m_i h_{n-k,i} &= 1
\end{aligned} \tag{5.7}$$

To satisfy the first inequality in eqn. (5.6) we require,

$$\sum_{i=1}^n m_i h_{z,i} = 1 \tag{5.8}$$

for some  $z = 1, 2, \dots, n - k$ .

Now if  $h_{z,i} = 0$  for all  $i = RS_{min} + 2, RS_{min} + 3, \dots, n$  then eqn. 5.8 becomes,

$$\sum_{i=1}^{RS_{min}+1} m_i h_{z,i} = 1 \tag{5.9}$$

So if we select  $m_1 \dots m_{RS_{min}+1}$  such that eqn. (5.9) is satisfied we know that the first inequality in eqn. (5.6) and hence the bound on  $RS_{min}$  is satisfied irrespective of what we select as values for

$$m_{RS_{min}+2} \dots m_n.$$

Consider now the second inequality in eqn. (5.6). If there exists some row  $z$  of  $H$  so that  $h_{z,i} = 0$  for all  $i = 1, MinRL_{max} + 3, MinRL_{max} + 4, \dots, n$ . i.e.

$$m_1 + \sum_{i=MinRL_{max}+3}^n m_i h_{z,i} = 1 \tag{5.10}$$

then the first inequality in eqn. (5.6) is satisfied simultaneously with the second inequality in eqn. (5.6) if we select  $m_1 \dots m_{MinRL_{max}+2}$  such that  $m_1 \dots m_{RS_{min}+1}$

satisfies eqn. (5.9) and  $m_2 \cdots m_{MinRL_{max}+2}$  satisfies eqn. (5.10). This allows  $m$  to satisfy the bound on  $RS_{min}$  and also on  $MinRL_{max}$  over the first  $MinRL_{max} + 2$  bits of  $m$ .

Thus in general to simultaneously satisfy the set of inequalities in eqn. (5.6) it suffices to have at least one row  $z$  of  $H$  so that,

$$h_{z,i} = 0 \text{ for all } i = RS_{min} + 2, RS_{min} + 3, \dots, n \quad (5.11)$$

at least one row so that,

$$h_{z,i} = 0 \text{ for all } i = 1, 2, \dots, p, MinRL_{max} + p + 2, MinRL_{max} + p + 3, \dots, n \quad (5.12)$$

where  $p = 1, 2, \dots, n - MinRL_{max} - 2$  and at least one row so that,

$$h_{z,i} = 0 \text{ for all } i = 1, 2, \dots, n - RE_{min} - 1 \quad (5.13)$$

Hence to find a modification vector which satisfies the predetermined runlength bound it suffices to perform row operations on  $H$  and to form the matrix  $S$  from the rows satisfying the conditions above. Then solving the matrix equation  $mS^t = 1$  yields a solution. Hence the algorithm for finding  $m$ .  $\square$

This algorithm is implemented in software and the  $\tilde{H}$  is obtained from the  $H$  matrix.

## 5.3 Examples

### Example 5.3.1 Computation of runlengths.



Given is the generator matrix  $G$  :

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

$RS_{min} = 1$  from first and second column

$RM_{min} = 2$ , from third and fourth column

$RE_{min} = 1$  from seventh and eighth column and

$$RL_{max} = \text{MAX}(1 + 1, 2) = 2$$

**Step 1: Find parity check matrix  $H$ .**

The systematic  $G_{sys}$  matrix is obtained by column permutations.

$$G_{sys} = \begin{matrix} & 1 & 2 & 3 & 5 & 7 & 4 & 6 & 8 \\ \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

The systematic  $H_{sys}$  matrix is obtained from  $G_{sys}$ .

$$H_{sys} = \begin{matrix} & 1 & 2 & 3 & 5 & 7 & 4 & 6 & 8 \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \end{matrix}$$

The parity check matrix  $H$  is obtained by performing reverse column permutations on the systematic matrix  $H_{sys}$

$$H = \begin{matrix} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \end{matrix}$$

By adding row #2 to row# 1:

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

By shifting row #2 to last row:

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

**Step 2: Conditions**

1.  $\tilde{H}$  has at least one such that

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 3, 4, \dots, 8$$

Row#1 of the  $H$  matrix satisfies this condition

2.  $\tilde{H}$  has at least one row  $z$  such that:  $H_{z,i} = 0$  for  $p = 1, 2, \dots, (8 - 2 - 2) = 4$ .

Thus  $p = 1, 2, 3, 4$

- (a) For  $p=1$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1 \text{ and } i = 5, 6, 7, 8$$

Row2 of the  $H$  matrix satisfies this condition

- (b) For  $p=2$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2 \text{ and } i = 6, 7, 8$$

Row2 of the  $H$  matrix satisfies this condition

(c) For  $p=3$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, 3 \text{ and } i = 7, 8$$

Row3 of the  $H$  matrix satisfies this condition

(d) For  $p=4$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, 3, 4 \text{ and } i = 8$$

Row3 of the  $H$  matrix satisfies this condition

3.  $\tilde{H}$  has at least one row  $z$  such that:

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, \dots, 6$$

Row4 of the  $H$  matrix satisfies this condition.

**Step 3 Find  $\tilde{H}$ .**

Since all the conditions of Step 2 are satisfied, no adjustment of bounds is needed.

Thus  $\tilde{H}$  is same as  $H$ .

**Step 4 Find  $s$ .**

Row 5 is redundant, thus matrix  $S$  is formed by eliminating the fifth row from the parity check matrix  $H$ .

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

**Step 5 Solving the equation  $mS^t = 1$ .**

*The modification vector  $m = (x_1x_2x_3x_4x_5x_6x_7x_8)$  must satisfy:*

$$x_1 + x_2 = 1 \Rightarrow (x_1 = 0 \& x_2 = 1) \text{ or } (x_1 = 1 \& x_2 = 0)$$

$$x_3 + x_4 = 1 \Rightarrow (x_3 = 0 \& x_4 = 1) \text{ or } (x_3 = 1 \& x_4 = 0)$$

$$x_5 + x_6 = 1 \Rightarrow (x_5 = 0 \& x_6 = 1) \text{ or } (x_5 = 1 \& x_6 = 0)$$

$$x_7 + x_8 = 1 \Rightarrow (x_7 = 0 \& x_8 = 1) \text{ or } (x_7 = 1 \& x_8 = 0)$$

*There are  $2^4 = 16$  different solutions. One of the possible solution is:*

$$m = [10101010]$$

*Thus modification vector  $m$  for the given code is given by  $m = [10101010]$*

**Example 5.3.2** *Given the generator matrix  $G$ :*

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

$RS_{min} = 1$ , from first and second column.

$RM_{min} = 3$ , from third, fourth, fifth and sixth column.

$RE_{min} = 3$ , from the last three columns and

$$RL_{max} = \text{MAX}(1 + 3, 3) = 4$$

Step 1: Find parity check matrix  $H$ .

By column permutations of the  $G$ , the systematic matrix  $G_{sys}$  is obtained:

$$G_{sys} = \begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 5 & 7 & 9 & & 4 & 6 & 8 & 10 & 11 & 12 \end{array} \\ \left[ \begin{array}{cccccccccccc} 1 & 1 & 1 & 0 & 0 & 0 & & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 & 1 & & 0 & 0 & 0 & 0 & 0 & 1 \end{array} \right] \end{array}$$

$H_{sys}$  is obtained from  $G_{sys}$ :

$$H_{sys} = \begin{array}{c} \begin{array}{cccccccccccc} 1 & 2 & 3 & 5 & 7 & 9 & & 4 & 6 & 8 & 10 & 11 & 12 \end{array} \\ \left[ \begin{array}{cccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & & 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & & 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & & 0 & 0 & 0 & 1 & 1 & 1 \end{array} \right] \end{array}$$

After reversing the column permutations:

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

Adding row #2 to row #1 and row #4 to row #3:

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

Arrange the rows to obtain the following parity check matrix  $H$ :

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

**Step 2: Conditions**

1. There must be at least one row so that:  $RS_{min} + 2 = 1 + 2 = 3$ . Thus

$$i = 3, 4, \dots, 12$$

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 3, 4, \dots, 12$$

Row1 of the H matrix satisfies this condition

2.  $\tilde{H}$  should have at least one row so that:  $n - MinRL_{max} - 2 = 12 - 4 - 2 = 6$ .

$$\text{thus } p = 1, 2, \dots, 6, MinRL_{max} + p + 2 = 4 + p + 2 = 6 + p$$

(a) For  $p=1$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1 \text{ and } i = 7, 8, 9, 10, 11, 12$$

Row2 of the H matrix satisfies this condition.

(b) For  $p=2$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2 \text{ and } i = 8, 9, 10, 11, 12$$

Row2 of the H matrix satisfies this condition

(c) For  $p=3$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, 3 \text{ and } i = 9, 10, 11, 12$$

Row3 of the H matrix satisfies this condition

(d) For  $p=4$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, 3, 4 \text{ and } i = 10, 11, 12$$

Row3 of the H matrix satisfies this condition



(e) For  $p=5$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, 3, 4, 5 \text{ and } i = 11, 12$$

*Row3 of the H matrix satisfies this condition*

(f) For  $p=6$ :

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, 3, 4, 5, 6 \text{ and } i = 12$$

*Row3 of the H matrix satisfies this condition*

3.  $\tilde{H}$  has at least one row so that:

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, \dots, 6$$

*Row3 of the H matrix satisfies this condition*

**Step 3 Find  $\tilde{H}$ .**

*Since all the conditions of Step 2 are satisfied, no adjustment of bounds is needed.*

**Step 4 Find  $s$ .**

*The rows 4, 5 and 6 are redundant, thus matrix  $S$  is formed by eliminating the last three rows from the parity check matrix  $H$ .*

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

**Step 5 Solving the equation  $mS' = 1$ .**

*The modification vector*

$$m = (x_1 x_2 x_3 x_4 x_5 x_6 x_7 x_8 x_9 x_{10} x_{11} x_{12})$$

*must satisfy:*

$$x_1 + x_2 = 1 \quad \Rightarrow \quad 2 \text{ possible solutions}$$

$$x_3 + x_4 + x_5 + x_6 = 1 \quad \Rightarrow \quad 8 \text{ possible solutions}$$

$$x_7 + x_8 = 1 \quad \Rightarrow \quad 2 \text{ possible solutions}$$

$$x_9, x_{10}, x_{11}, x_{12} \in \{0, 1\} \quad \Rightarrow \quad 16 \text{ possible solutions}$$

*There are  $2 \times 8 \times 2 \times 16 = 512$  different solutions. One of the possible solution*

*is:*

$$m = [101000101000]$$

*Thus modification vector  $m$  for the given code is given by  $m = [101000101000]$*

## 5.4 Generator Matrix Transformation

Let  $G$  be a generator matrix of an  $(n, k)$  code over  $GF(2^m)$ . We want to replace every symbol of the extension field by its binary equivalent. Let  $(n_2, k_2)$  be the binary code equivalent to  $(n, k)$  where  $n_2 = n \times m$  and  $k_2 = k \times m$ , and let  $G_2$  be the generator matrix of the  $(n_2, k_2)$  code. There are  $k_2$  rows in  $G_2$ . The binary

generator matrix is simply obtained from the original matrix  $G$  of the  $(n, k)$  RS code over  $GF(2^m)$  as

$$G_2 = \begin{bmatrix} \alpha^0 G \\ \alpha^1 G \\ \alpha^2 G \\ \vdots \\ \alpha^{m-1} G \end{bmatrix}$$

where every (field) element of  $GF(2^m)$  in  $G$  is replaced by its binary row vector of length  $m$ . Thus  $G_2$  has dimension  $(m.k) \times (m.n) = k_2 \times n_2$ .

$$G_2 = \begin{bmatrix} \alpha^3 & \alpha & 1 & \alpha^3 & 1 & 0 & 0 \\ 0 & \alpha^3 & \alpha & 1 & \alpha^3 & 1 & 0 \\ 0 & 0 & \alpha^3 & \alpha & 1 & \alpha^3 & 1 \\ \hline \alpha^4 & \alpha^2 & \alpha & \alpha^4 & \alpha & 0 & 0 \\ 0 & \alpha^4 & \alpha^2 & \alpha & \alpha^4 & \alpha & 0 \\ 0 & 0 & \alpha^4 & \alpha^2 & \alpha & \alpha^4 & \alpha \\ \hline \alpha^5 & \alpha^3 & \alpha^2 & \alpha^5 & \alpha^2 & 0 & 0 \\ 0 & \alpha^5 & \alpha^3 & \alpha^2 & \alpha^5 & \alpha^2 & 0 \\ 0 & 0 & \alpha^5 & \alpha^3 & \alpha^2 & \alpha^5 & \alpha^2 \end{bmatrix}$$

Now convert every symbol of  $G_2$  to its binary equivalent.







After column permutations a systematic generator matrix is obtained:

$$G_{sys} = \begin{array}{c} \begin{array}{cccccccccccccccccccc} 21 & 20 & 19 & 18 & 4 & 16 & 3 & 10 & 8 & 1 & 2 & 5 & 6 & 7 & 9 & 11 & 12 & 13 & 14 & 15 & 17 \end{array} \\ \left[ \begin{array}{cccccccccccccccccccc} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \end{array} \right] \end{array}$$

The systematic  $H_{sys}$  matrix is obtained from the  $G_{sys}$ .

$$H_{sys} = \begin{matrix} & \begin{matrix} 21 & 20 & 19 & 18 & 4 & 16 & 3 & 10 & 8 & 1 & 2 & 5 & 6 & 7 & 9 & 11 & 12 & 13 & 14 & 15 & 17 \end{matrix} \\ \begin{matrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{matrix} \end{matrix}$$

The parity check matrix  $H$  is obtained by reversing the column permutations of



the systematic matrix  $H_{sys}$

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

Step 2: Conditions

1. Since  $RS_{min} = 9$ , there must be at least one row so that

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 11, 12, \dots, 21$$

This condition is not satisfied.

2. Since  $MinRL_{max} = 18$ , it follows that  $n - MinRL_{max} - 2 = 21 - 18 - 2 = 1$ .

Thus  $p = 1$ . Hence  $i = 1, 2, \dots, p, MinRL_{max} + p + 2, \dots, n = 1, 21$ .  $\tilde{H}$  should

have at least one row so that:

$$\tilde{h}_{z,1} = \tilde{h}_{z,21} = 0$$

Row2 of the  $H$  matrix satisfies this condition

3. Since  $MinRL_{end} = 18$ , it follows that  $n - RE_{min} - 1 = 21 - 9 - 1 = 11$ . Hence

$i = 1, 2, \dots, 11$ .  $\tilde{H}$  has at least one row so that:

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, \dots, 11$$

This condition is not satisfied.

**Step 3 Find  $\tilde{H}$ .**

The matrix  $H$  is modified to satisfy the conditions and its modification can be written as:



have at least one row so that:

$$\tilde{h}_{z,1} = \tilde{h}_{z,21} = 0$$

Row#2 of the  $\tilde{H}$  matrix satisfies this condition

3. Since  $MinRL_{end} = 18$ , it follows that  $n - RE_{min} - 1 = 21 - 9 - 1 = 11$ . Hence

$i = 1, 2, \dots, 11$ .  $\tilde{H}$  has at least one row so that:

$$\tilde{h}_{z,i} = 0 \text{ for all } i = 1, 2, \dots, 11$$

Row#11 of the  $\tilde{H}$  matrix satisfies this condition

**Step 4 Find  $S$ .**

The rows 3,4  $\dots$  10 and 12 are redundant, thus matrix  $S$  is formed by eliminating these rows from the parity check matrix  $\tilde{H}$ .

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

**Step 5 Solve the equation  $mS' = 1$**  Solving the equation  $mS' = 1$  gives many solutions. One of the possible solutions is

$$m = [100100000000100000000]$$

# Chapter 6

## REFLECTIVE CODES

In this chapter simpler implementation of some Reflective Codes are discussed. Simple methods for the parity-check matrix and modification vector are investigated. Several codes are given for illustration.

### 6.1 Parity-check Matrix Construction

Popplewell and O'Reilly [19] investigated a subclass of reflective codes, which we shall refer to as PO(Popplewell-O'Reilly) codes. These codes can be defined by specifying their  $H$  matrix.

**Definition 6.1.1** *Let  $i_1, i_2, m_1$ , and  $m_2$  be positive integers, satisfying*

1.  $i_1$  and  $i_2$  are even
2.  $i_1 \leq i_2$

3.  $(m_1 - m_2) \in \{0, 1\}$

4.  $(m_1 i_1 + m_2 i_2)/2$  is even

Furthermore, let  $m = m_1 + m_2$  and  $L = \log_2 [i_2]$ . Then we define the parity check matrix  $H$  of the Popplewell-O'Reilly (PO) code to be the  $(m + L) \times (m_1 i_1 + m_2 i_2)$  matrix consisting of alternating blocks  $B_{r_1}^1$  (of  $i_1$  columns) and  $B_{r_2}^2$  (of  $i_2$  columns) such that, for  $m_1 = m_2$ .

$$H = [B_0^1 B_1^2 B_2^1 B_3^2 \cdots B_{m-2}^1 B_{m-1}^2]$$

and for  $m_1 \neq m_2$

$$H = [B_0^1 B_1^2 B_2^1 B_3^2 \cdots B_{m-2}^2 B_{m-1}^1]$$

where

$$B_r^k = \begin{bmatrix} b_{2^r}^m & b_{2^r}^m & b_{2^r}^m & \cdots & b_{2^r}^m \\ b_0^k & b_1^k & b_2^k & \cdots & b_{i_k-1}^k \end{bmatrix}$$

$k \in \{1, 2\}$  and  $r \in \{0, 1, 2, \dots, m-1\}$

and  $b_j^k$  denotes a binary column vector with  $k$  entries and binary value  $j$ , when read from bottom (MSB) to top (LSB).

**Proposition 6.1.1** *The  $(n, k)$  PO code defined by the  $H$  matrix of Definition 6.1.1 satisfies the following properties:*

1.  $C$  is a reflective code
2.  $d_{\min}(C) = 4$

$$3. n = m_1 i_1 + m_2 i_2$$

$$4. k = m_1(i_1 - 1) + m_2(i_2 - 1) - L = n - (m + L). \text{ where } L = \lceil \log_2 i_2 \rceil$$

$$5. RL_{max} = i_1 + i_2 - 2$$

**Proof:**

(1) First  $m = m_1 + m_2$  rows of  $H$  contain  $i_1$  and  $i_2$  number of 1s alternately. Since  $i_1$  and  $i_2$  are even, it implies even number of 1s in the first  $m$  rows. The number of 1s in rows  $m + j, j = 1, 2, \dots, L$  is

$$m_1 2^{j-1} \left\lfloor \frac{i_1}{2^j} \right\rfloor + m_2 2^{j-1} \left\lfloor \frac{i_2}{2^j} \right\rfloor$$

This is clearly always even when  $j \geq 1$ , and when  $j = 1$  this expression becomes  $(m_1 i_1 + m_2 i_2)/2$ . This is also even. So every row of  $H$  matrix has an even number of 1s and hence the code is transparent.

(2) Since all the columns of the  $H$  matrix are different, then any three columns are independent and one deduces  $d_{min} \geq 3$ . The number of columns is always  $\geq 4$ , where equality holds for  $i_1 = i_2 = 2$ , and  $m_1 = m_2 = 1$ . Since the modulo-2 addition of the first four columns is zero, we have  $d_{min} \leq 4$ .

(3) Since  $i_1$  number of columns are repeated  $m_1$  times and  $i_2$  number of columns are repeated  $m_2$  times,  $n = m_1 i_1 + m_2 i_2$ .

(4) There are  $m_1 + m_2 + L$  rows in the H matrix. It implies that  $n - k = m_1 + m_2 + L$ . Thus  $k$  can be written as  $k = n - m_1 - m_2 - L$ . Substituting the value for  $n$ , we get  $k = m_1 i_1 + m_2 i_2 - m_1 - m_2 - L$ . Thus  $k = i_1(m_1 - 1) + i_2(m_2 - 1) - L$ .

(5) Since  $RS = i_1 - 1$  and  $RE = i_2 - 1$ , it implies that  $RL_{max} = RS + RE = i_1 + i_2 - 2$ . □

Note that  $i_1$  and  $i_2$  influence the runlengths of the modified codes while  $m_1$  and  $m_2$  influence the code rate.

The first  $m$  rows of H ensure that an appropriately modified code will satisfy the maximum runlength value above, while the last  $L$  rows ensure that the code has minimum distance 4.

## 6.2 Modification Vector

**Proposition 6.2.1** *The modification vector which maps the linear Popplewell-O'Reilly (PO) code to a coset satisfying  $RL_{max}$  is:*

for  $m_1 = m_2$  :

$$M = [\overbrace{10 \dots 0}^{i_1} \overbrace{10 \dots 0}^{i_2} \dots \overbrace{10 \dots 0}^{i_1} \overbrace{10 \dots 0}^{i_2}]$$

and for  $m_1 \neq m_2$  :

$$M = [\overbrace{10 \dots 0}^{i_1} \overbrace{10 \dots 0}^{i_2} \overbrace{10 \dots 0}^{i_1} \overbrace{10 \dots 0}^{i_2} \dots \overbrace{10 \dots 0}^{i_2} \overbrace{10 \dots 0}^{i_1}]$$



### 6.3 Generator Matrix

If the parity check matrix is given then the generator matrix could easily be obtained by first changing the parity check matrix to the systematic parity check matrix and then from systematic parity check matrix we obtain the systematic generator matrix. From the systematic generator matrix we obtain the required generator matrix. The conversion of the parity check matrix to the systematic parity check matrix may need row operations. But row operations do not change the code. If column permutations are also required for obtaining systematic parity check matrix then the generator matrix obtained would not be identical to the desired generator matrix. This generator matrix obtained after column permutations would be equivalent to the desired generator matrix. To get the desired generator matrix inverse column operations should be performed.

### 6.3.1 Generator Matrix of Subclass of Reflective Codes

For the special case of reflective codes for which  $i_1 = i_2 = 2$ , the generator matrix is

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

where, as usual,  $G$  is a  $k \times n$  matrix

**Proposition 6.3.1** *When  $i_1 = i_2 = 2$  then  $m_1 = m_2$ .*

**Proof:** Let  $i_1 = i_2 = i$ . Then  $m_1 i_1 + m_2 i_2 = 2m_1 + 2m_2 = 2(m_1 + m_2)$ . But  $(m_1 i_1 + m_2 i_2)/2$  should be even. This implies that  $m_1 + m_2$  should be even. But  $m_1 - m_2 \in \{0, 1\}$  implies that  $m_1 = m_2$  ;i.e.,  $m_1 - m_2 = 0$  otherwise, if  $m_1 - m_2 = 1$  then  $m_1 + m_2 = m_2 + m_2 + 1 = 2m_2 + 1$ . This number is odd in all cases because  $2m_2 + 1$  is odd for all cases of  $m_2$ .  $\square$

## 6.4 Software

Software modules have been written to generate parity check matrix and generator matrix for a code with the given parameters of  $i_1, i_2, m_1$  and  $m_2$ .

To avoid the difficulty of getting the  $G$  matrix from  $H$  by solving  $GH^T = 0$ , the cyclic-like, well structured and systematic nature of the matrices was exploited to obtain a software function for generating  $G$ .

The systematic  $H$  matrix is obtained after column permutation of the  $H$  matrix. Then the systematic  $G$  matrix is obtained from the systematic  $H$  matrix. From this systematic  $G$  matrix, the actual  $G$  matrix is obtained by reverse column permutation. See the flow chart in figure (6.1).

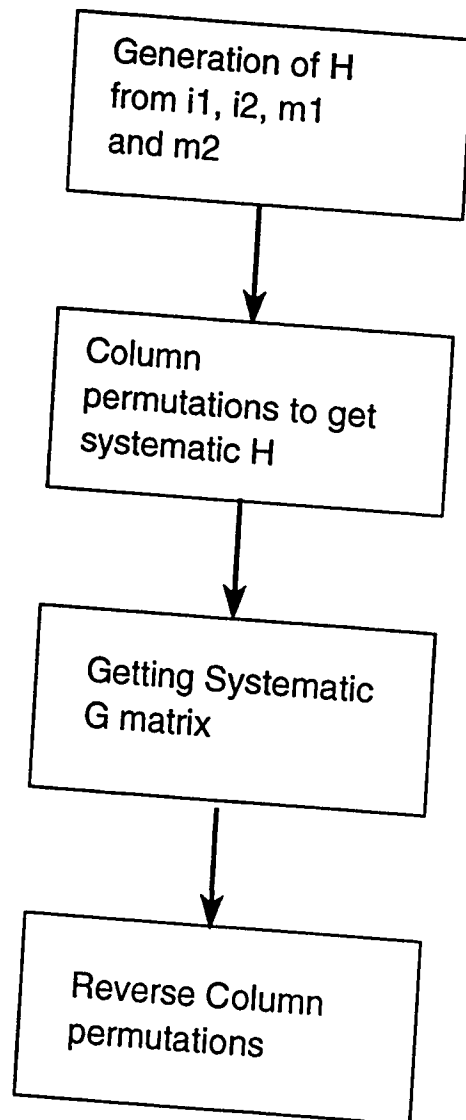


Figure 6.1: Flowchart for obtaining G matrix

## 6.5 Examples

### 6.5.1 Example #1

$$i_1 = 2 \qquad i_2 = 2$$

$$m_1 = 2 \qquad m_2 = 2$$

$$n = 8 \qquad k = 3$$

$$m = m_1 + m_2 = 4$$

$$L = \lceil \log_2 2 \rceil = 1$$

$$B_r^1 = \begin{bmatrix} b_{2r}^4 & b_{2r}^4 \\ b_0^1 & b_1^1 \end{bmatrix}$$

Since  $i_1 = i_2$ ,  $B_r^1 = B_r^2$ , where  $r \in \{0, 1, 2, 3\}$

$$H_1 = [B_0^1 B_1^2 B_2^1 B_3^2]$$

$$H_1 = \begin{bmatrix} b_1^4 & b_1^4 & b_2^4 & b_2^4 & b_4^4 & b_4^4 & b_8^4 & b_8^4 \\ b_0^1 & b_1^1 & b_0^1 & b_1^1 & b_0^1 & b_1^1 & b_0^1 & b_1^1 \end{bmatrix}$$

$$H_1 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Total number of words = 256

Number of codewords = 8

Number of cosets = 32

Maximum Runlength,  $RL_{max} = i_1 + i_2 - 2 = 2 + 2 - 2 = 2$

Modification Vector = 10101010

### 6.5.2 Example #2

$$i_1 = 2 \qquad i_2 = 2$$

$$m_1 = 3 \qquad m_2 = 3$$

$$n = 12 \qquad k = 5$$

$$m = m_1 + m_2 = 6$$

$$L = \lceil \log_2 2 \rceil = 1$$

$$B_r^1 = \begin{bmatrix} b_{2r}^6 & b_{2r}^6 \\ b_0^1 & b_1^1 \end{bmatrix}$$

Since  $i_1 = i_2$ ,  $B_r^1 = B_r^2$ , where  $r \in \{0, 1, 2, 3, 4, 5\}$

$$H_2 = [B_0^1 B_1^2 B_2^1 B_3^2 B_4^1 B_5^2]$$

$$H_2 = \begin{bmatrix} b_1^6 & b_1^6 & b_2^6 & b_2^6 & b_4^6 & b_4^6 & b_8^6 & b_8^6 & b_{16}^6 & b_{16}^6 & b_{32}^6 & b_{32}^6 \\ b_0^1 & b_1^1 & b_0^1 & b_1^1 & b_0^1 & b_1^1 & b_0^1 & b_1^1 & b_0^1 & b_1^1 & b_0^1 & b_1^1 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}$$

$$G_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Total number of words = 4096

Number of codewords = 32

Number of cosets = 128

Modification Vector = 1010101010

Maximum Runlength,  $RL_{max} = i_1 + i_2 - 2 = 2 + 2 - 2 = 2$

### 6.5.3 Example # 3

$$i_1 = 2 \qquad i_2 = 4$$

$$m_1 = 2 \qquad m_2 = 2$$

$$n = 12 \qquad k = 6$$

$$m = m_1 + m_2 = 4$$

$$L = \lceil \log_2 4 \rceil = 2$$

$$B_r^1 = \begin{bmatrix} b_{2r}^4 & b_{2r}^4 \\ b_0^2 & b_1^2 \end{bmatrix}$$

$$B_r^2 = \begin{bmatrix} b_{2r}^6 & b_{2r}^6 & b_{2r}^6 & b_{2r}^6 \\ b_0^2 & b_1^2 & b_2^2 & b_3^2 \end{bmatrix}$$

where  $r \in \{0, 1, 2, 3\}$

$$H_3 = [B_0^1 B_1^2 B_2^1 B_3^2]$$

$$H_3 = \begin{bmatrix} b_1^4 & b_1^4 & b_2^4 & b_2^4 & b_2^4 & b_2^4 & b_4^4 & b_4^4 & b_8^4 & b_8^4 & b_8^4 & b_8^4 \\ b_0^2 & b_1^2 & b_0^2 & b_1^2 & b_2^2 & b_3^2 & b_0^2 & b_1^2 & b_0^2 & b_1^2 & b_2^2 & b_3^2 \end{bmatrix}$$



$$H_3 = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

$$G_3 = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Total number of words = 4096

Number of codewords = 64

Number of cosets = 64

Maximum Runlength,  $RL_{max} = i_1 + i_2 - 2 = 2 + 4 - 2 = 4$

Modification Vector = 101000101000

# Chapter 7

## WEIGHT DISTRIBUTION

### 7.1 Weight Enumerator

**Definition 7.1.1** *If  $K$  is a block code with exactly  $A_i$  codewords of Hamming weight  $i$  ( $i=0,1,\dots,n$ ) then the polynomial*

$$A(x) = \sum_{i=0}^n A_i x^i$$

*is called the 'weight enumerator' or 'weight distribution' of the code  $K$ .*

For any small code, we can find a table of all the weights by an exhaustive search.

For large codes, this is not possible. Instead, one must employ analytical techniques if such techniques can be found in general. Analytically describing the weight distribution of a code is a difficult problem and is unsolved for most codes.

**Proposition 7.1.1** *If the minimum distance is  $d$ , then  $A_0 = 1$ ,  $A_1 = A_2 = \dots = A_{d-1} = 0$ .*

... $A_{d-1} = 0$ ,  $A_d$  not zero. Furthermore, if  $C$  is transparent then

$$A_i = A_{n-i} \text{ for } i = 0, 1, \dots, \lfloor n/2 \rfloor$$

**Proof:** The weight of a codeword is its distance from the all zero word. If the minimum distance is  $d$ , then there is no codeword at distance less than  $d$  from the all-0 codeword. Thus all the codewords are at distance greater or equal to  $d$  from the all-zero codeword. The distance of the non-zero codeword from the all-zero codeword is the Hamming weight of the codeword. Thus all the non-zero codewords are of weight greater or equal to  $d$ . Hence the constants  $A_1, \dots, A_{d-1}$  are all zero.

If a transparent code contains a codeword of weight  $i$ , then the corresponding complement codeword has weight  $(n - i)$ . If there are  $A_i$  number of codewords of weight  $i$  then there must be  $A_{n-i}$  codewords of weight  $(n - i)$ .  $\square$

**Example 7.1.1** For the Hamming code  $(7,3)$ .  $A_3 = A_4 = 7$ , for the Hamming code  $(15,11)$ ,  $A_3 = A_{12} = 35$ ,  $A_5 = A_{10} = 168$ .

## 7.2 Weight Distribution of Maximum Distance

### Separable (MDS) codes

**Definition 7.2.1** An  $(n,k)$  code with minimum distance  $d = n - k + 1$  is called Maximum Distance Separable (MDS).

Maximum distance separable codes are defined over fields  $GF(q)$  with  $q > 2$ . Since these codes can have more than one codewords of Hamming weight equal to  $n$ , it follows that the number of codewords of weight 0 is not equal to number of codewords of weight  $n$ . Thus these codes cannot be considered for transparent codes. The known weight distribution of MDS codes does not exclude the possibility that an MDS code is transparent.

**Proposition 7.2.1** *The weight distribution of a MDS code  $(n,k)$  over  $GF(q)$  is given by*

$$A_l = \begin{cases} 1 & \text{if } l = 0 \\ 0 & \text{if } l = 1, \dots, (d-1) \\ \binom{n}{l} (q-1) \sum_{j=0}^{l-d} (-1)^j \binom{w-1}{j} q^{l-d-j} & \text{if } l \geq d \end{cases}$$

where  $d$  is the minimum distance.

**Proof:** For proof see [40]. □

### 7.3 Weight Distribution of Hamming-Codes

**Proposition 7.3.1** *The weight distribution of the class of Hamming  $(n,k)$  codes is*

:

$$A(x) = \frac{1}{n+1} \left[ (1+x)^n + n(1+x)^{(n-1)/2} (1-x)^{(n+1)/2} \right]$$

**Proof:** For proof see [40] □

The weight coefficients for various Hamming codes are listed in Tables 7.1 and

7.2.

Table 7.1: Weight Distribution of Hamming Code(n,k) = (7,4)

<i>Weight</i>	No. of equiweight <i>Codewords</i>
0	1
3	7
4	7
7	1

Table 7.2: Weight Distribution of Hamming Code  $(n,k) = (15,11)$ 

<i>Weight</i>	No. of equiweight <i>codewords</i>
0	1
3	35
4	105
5	168
6	280
7	435
8	435
9	280
10	168
11	105
12	35
15	1

## 7.4 Examples

$$i_1 = 2 \quad i_2 = 2$$

**Example 7.4.1**  $m_1 = 3 \quad m_2 = 3$  *Modification Vector = 1010101010*

$$n = 12 \quad k = 5$$

*Weight Distribution: (Original Code)*

*Weight* 0 4 8 12

*Number* 1 15 15 1

*Weight Distribution: (Modified Code)*

*Weight* 6

*Number* 32

*See figure 7.1.*



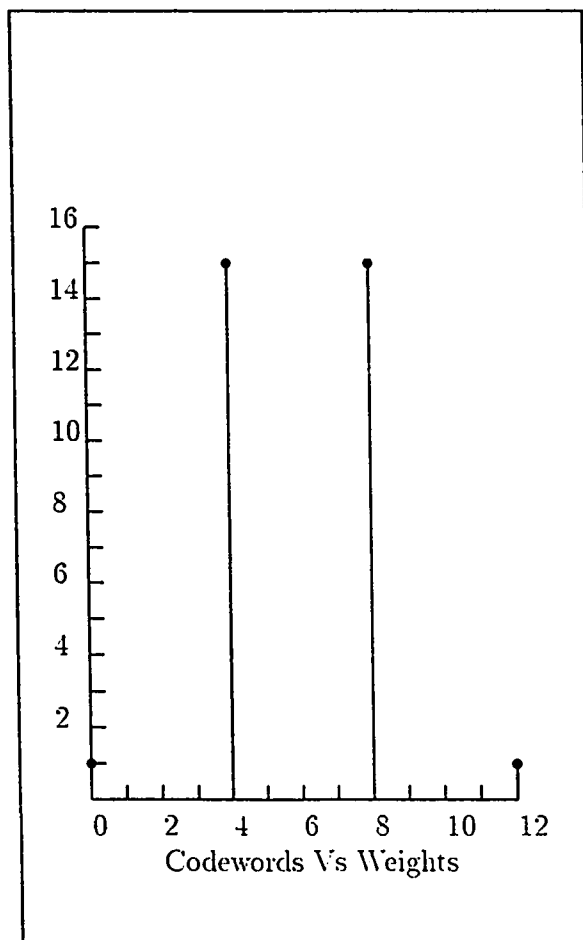


Figure 7.1: Weight distribution of (12,5) PO code

$$i_1 = 2 \quad i_2 = 4$$

**Example 7.4.2**  $m_1 = 2 \quad m_2 = 2$  *Modification Vector = 101000101000*

$$n = 12 \quad k = 6$$

*Weight Distribution: (Original Code)*

*Weight* 0 4 6 8 12

*Number* 1 23 16 23 1

*Weight Distribution: (Modified Code)*

*Weight* 4 6 8

*Number* 16 32 16

*See figure 7.2.*

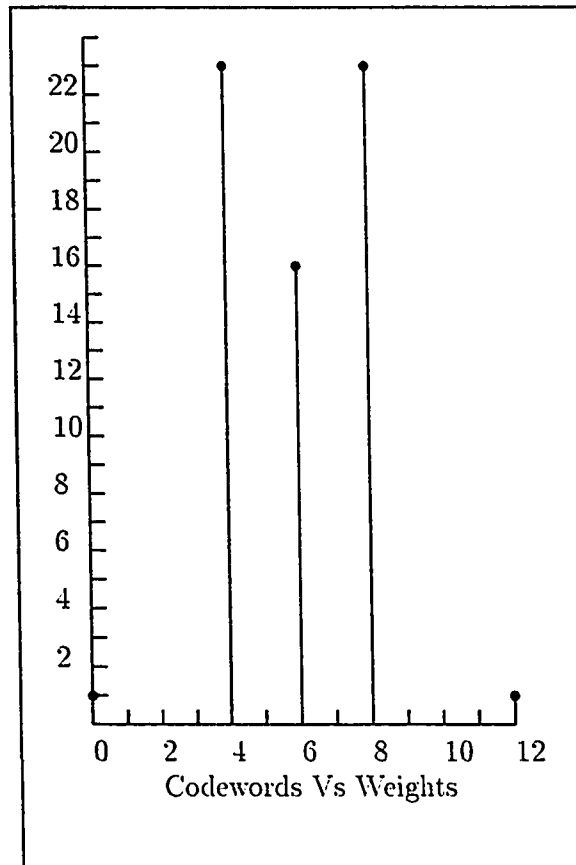


Figure 7.2: Weight distribution of (12,6) PO code

## 7.5 Weight Distribution of Subclass of Reflective Codes

To find a general formula for the weight distribution of transparent codes, consider first special subclasses.

**Proposition 7.5.1** *For the subclass of PO reflective codes with  $i_1 = i_2 = 2$ , the weight distribution is given by:*

$$A(x) = x^0 + \sum_{i=1}^{(k-1)/2} \left( \binom{k}{2i-1} + \binom{k}{2i} \right) x^{4i} + x^n$$

**Proof:** Since the weight enumerator  $A_i = A_{n-i}$  and  $A_0 = A_1 = 1$  are satisfied in all the examples, thus the weight distribution is satisfying the requirement of transparent codes over  $GF(2)$ . If all the codewords are added then the expression becomes:

$$\begin{aligned} \# \text{ of codewords} &= \binom{k}{0} + \binom{k}{1} + \binom{k}{2} + \cdots + \binom{k}{k-1} + \binom{k}{k} \\ 2^k &= \binom{k}{0} + \binom{k}{1} + \binom{k}{2} + \cdots + \binom{k}{k-1} + \binom{k}{k} \end{aligned}$$

Since the summation of all coefficients of the weight enumerator is equal to the total number of codewords, it implies that the distribution does not contradict with the known facts. □

See tables 7.3, 7.4, and 7.5.

Table 7.3: Weight Distribution  $(n,k) = (20,9)$ 

<i>Code</i>	<i>Weight</i>	Codewords of Equivalent Weight
$i_1 = 2$	0	1
$i_2 = 2$	4	45
$m_1 = 5$	8	210
$m_2 = 5$	12	210
$n = 20$	16	45
$k = 9$	20	1

Table 7.4: Weight Distribution  $(n,k) = (20,11)$ 

<i>Code</i>	<i>Weight</i>	<i>Codewords Of Equivalent Weight</i>
$i_1 = 2$	0	1
$i_2 = 4$	4	69
$m_1 = 4$	6	144
$m_2 = 3$	8	570
$n = 20$	10	480
$k = 11$	12	570
	14	144
	16	69
	20	1

Table 7.5: Weight Distribution  $(n,k) = (32,15)$ 

<i>Code</i>	<i>Weight</i>	<i>Codewords Of Equivalent Weight</i>
$i_1 = 2$	0	1
$i_2 = 2$	4	120
$m_1 = 8$	8	180
$m_2 = 8$	12	8008
$n = 32$	16	12870
$k = 15$	20	8008
	24	1820
	28	120
	32	1

## 7.6 Software

Computer search has been applied to get the weight distributions of various smaller codes. The program produced sufficient data to obtain the weight distribution of several codes. See table 7.6.



Table 7.6: Weight Distribution  $(n,k) = (32,19)$ 

<i>Code</i>	<i>Weight</i>	Codewords Of Equivalent Weight
$i_1 = 2$	0	1
$i_2 = 4$	4	200
$m_1 = 6$	6	960
$m_2 = 5$	8	7740
$n = 32$	10	22720
$k = 19$	12	65528
	14	99200
	16	131590
	18	99200
	20	65528
	22	22720
	24	7740
	26	960
	28	200
	32	1

## Chapter 8

# CONCLUSIONS AND FUTURE RESEARCH

Runlength-limited codes have wide applications in storage devices, optical discs, Digital Audio Tape (DAT), CD's. Runlength constraints are necessary to be satisfied. Low frequency components are normally used for control signals and servo mechanisms. If transitions of information bits is quite rapid then intersymbol interference may be the problem.

Separate codes could be used for error correction and runlength constraints. But the codes performing error correction and satisfying the runlength constraints are far more efficient than the concatenation scheme.

Transparent codes are self complementary. Every codeword contains its complement in the dictionary of the codewords. Runlength could be infinite in case of

transparent codes.

Many popular codes are found to be transparent. Hamming codes, repetition code, perfect code, Reed Solomon Code are transparent. General properties of transparent codes are investigated.

A new class of transparent codes has been studied. Parity-check matrix and generator matrix of subclass of transparent codes have been implemented in software. These codes are quite simple for implementation. The encoders and decoders could be implemented quite efficiently.

Linear codes have been modified to satisfy the runlength constraints. Error correction capabilities of the codes are preserved after modification. This modification is performed over binary and binary extension fields. Different modification vectors can be used for a code depending on our application.

# Bibliography

- [1] Immink K.A.S. *Coding Techniques for Digital Recorder*. Prentice Hall International (UK), 1991.
- [2] J.G. Proakis. *Digital Communications (Second Edition)*. McGraw-Hill International Editions, 1989.
- [3] J. Ashley. "Sliding Block Codes Between Constrained Systems". *IEEE Transactions on Information Theory*, IT-39(4):1303-1309, July 1993.
- [4] P. H Siegel. "Recording codes for digital magnetic storage". *IEEE Transactions on Magnetics*, MAG-21(5):1344-1349, September. 1985.
- [5] T.D. Howell. "Analysis of Correctable Errors in the IBM 3380 Disk File". *IBM Journal of Research and Development*, 28(2):206-211, March 1984.
- [6] M. Berkoff. "Waveform Compression in NRZI Magnetic Recording". *IEEE Proceedings*, 52(10):1271-1272, October 1964.

- [7] C.V. Freiman and A.D. Wyner. "Optimum Block Codes for Noiseless Input Restricted Channels". *Information and Control*, 7:398-415, 1964.
- [8] W.H. Kautz. "Fibonacci Codes for Synchronization Control". *IEEE Transaction of Information Theory*, IT-11:284-292, 1965.
- [9] A. Gabor. "Adaptive Coding for Self-Clocking Recording". *IEEE Transactions Electronic Computers*, EC-16(12):866-868, December 1967.
- [10] D.T. Tang and L.R. Bahl. "Block Codes for a Class of Constrained Noiseless Channels". *Information and Control*, 17:436-461, 1970.
- [11] P.A. Franaszek. "Sequence-State Encoding for Digital Transmission". *Bell Systems Technical Journal*, 47(1):143-157, January 1968.
- [12] P.H. Siegel and J.K. Wolf. "Modulation and Coding for Information Storage". *IEEE Communications Magazine*, December 1991.
- [13] K.A.S. Immink. "Runlength Limited Sequences". *IEEE Proceedings*, 78(11):1745-1759, November 1990.
- [14] B.H. Marcus, P.H. Siegel, and J.K. Wolf. "Finite-State Modulation Codes for Data Storage". *IEEE Journal on Selected Areas in Communications*, SAC-10(1):5-37, January 1992.
- [15] K.C. Pohlmann. *Advanced Digital Audio*. SAMS. 1991.

- [16] A. Popplewell and J.J. O'Reilly. "New class of runlength-limited error-control codes with minimum distance 4 ". *IEE Proceedings-I*, 140(2):104–108, April 1993.
- [17] M. Blaum. "Combining ECC with modulation: performance comparisons ". *IEEE Transactions on Information Theory*, IT-37(3):945–948, May 1991.
- [18] A. Kokos, J.J. O'Reilly, Popplewell A., and S. Williams. "Evaluation of a Class of Error Control Line Codes: an error performance perspective". *IEE Proceedings Part I*, 139(2):128–132, April 1992.
- [19] Popplewell A. and J.J. O'Reilly. "Runlength Limited Binary Error Control Codes". *IEE Proceedings Part I*, 139(3):349–355. June 1992.
- [20] H.C. Ferreira and S. Lin. "Error and Erasure Control (d,k) Block Codes". *IEEE Transactions on Information Theory*, IT-37(5):1399–1408, September 1991.
- [21] A.S.J. Helberg and H.C Ferreira. "Some New Runlength Constrained Binary Modulation Codes with Error-Correcting Capabilities". *Electronics Letters*. 28(2):137–139, January 1992.
- [22] H.C. Ferreira, D.A. Wright, and A.L. Nel. "Hamming Distance Preserving Mappings and Trellis Codes with Constrained Binary Symbols". *IEEE Transactions on Information Theory*, IT-35:1098–1103, September 1989.

- [23] L.J. Fredrickson and J.K. Wolf. "Error Detecting Multiple Block (d,k) Codes". *IEEE Transactions on Magnetics*, MAG-25:4096–4098, September 1989.
- [24] P. Lee and J.K. Wolf. "A General Error-Correcting Code Construction for Run-length Limited Binary Channels". *IEEE Transactions on Information Theory*, IT-35(6):1330–1335, November 1989.
- [25] K.A.S. Immink. "Coding Techniques for the Noisy Magnetic Recording Channel: A State-of-the-Art Report". *IEEE Transactions Communications*, COM-37(5):413–419, May 1989.
- [26] A. Patapoutian and P.V. Kumar. "The (d,k) Subcode of a Linear Block Code". *IEEE Transactions on Information Theory*. IT-38(4):1375–1382, July 1992.
- [27] Oyvind Ytrehus. "Runlength-Limited Codes for Mixed-Error Channels". *IEEE Transactions on Information Theory*, IT-37(6):1577–1585, November 1991.
- [28] A. Bassalygo. "Correcting Codes With an Additional Property". *Prob. Inform. Transm.*, 4(1):1–5, Spring 1968.
- [29] O. Ytrehus. "Upper Bounds on Error-Correcting Runlength-Limited Block Codes". *IEEE Transactions on Information Theory*, IT-37(3):941–945, May 1991.
- [30] K.A.S. Immink. "Construction of Almost Block Decodable Runlength-limited Codes". *IEEE Transaction on Information Theory*, 40(3), June 1994.

- [31] A.E. Brouwer, J.B. Shearer, N.J.A. Sloane, and W.D. Smith. “A New Table of Constant Weight Codes”. *IEEE Transactions on Information Theory*, IT-36(6):1334–1380, November 1990.
- [32] G.L. Pierobon. “Codes for Zero Spectral Density at Zero Frequency”. *IEEE Transactions on Information Theory*, IT-30(2):435–439, March 1984.
- [33] J. Justesen. “Information Rate and Power Spectra of Digital Codes”. *IEEE Transactions on Information Theory*, IT-28(3):457–472, May 1982.
- [34] K.A.S. Immink. “Performance of Simple Binary DC-Constrained Codes”. *Phillips Journal Research*, 40:1–21, 1985.
- [35] M. Blaum, S. Litsyn, V. Buskens, and H.C.A. Tilborg. “Error-Correcting Codes with Bounded Running Digital Sum”. *IEEE Transactions on Information Theory*, IT-39(1):216–227, January 1993.
- [36] G.D. Cohen and S.N. Litsyn. “DC-constrained Error-Correcting Codes with Small Running Digital Sum”. *IEEE Transactions on Information Theory*, IT-37(3):949–955, May 1991.
- [37] R.H. Deng, Y.X. Li, and M.A. Herro. “DC-Free Error-Correcting Convolutional Codes”. *Electronics Letters*, 29(22):1910–1911. October 1993.
- [38] A. Barg. “Incomplete Sums, DC-Constrained Codes, and Codes that Maintain Synchronization”. *Designs, Codes and Cryptography*, 3(1):105–116, 1993.



- [39] A.M. Barg and S.N. Litsyn. “DC-Constrained Codes from Hadamard Matrices”. *IEEE Transactions on Information Theory*, IT-37(3):801–807, May 1991.
- [40] F.J. MacWilliams and N.J.A. Sloane. *The Theory of Error-Correcting Codes*. North-Holland Publishing Company, 1977.
- [41] R.E. Blahut. *Theory and Practice of Error Control Codes*. Addison-Wesley Publishing Company, 1984.
- [42] J. Adamek. *Foundations of Coding*. John Wiley & Sons, Inc., 1991.

# Vitae

- Sajid Hussain
- Born in 1967 at Lahore, Pakistan.
- Received Bachelor of Engineering (B.E.) degree in Electrical Engineering from University of Engineering and Technology, Lahore, Pakistan in 1991.
- Received Diploma in Computer Science from National College of Computer Sciences, Lahore, Pakistan in 1992.
- Received Master of Science (M.S.) degree in Electrical Engineering from KFUPM, Saudi Arabia in 1995.