A HIGH RATE CODING SCHEME FOR
BYTE-ORIENTED INFORMATION SYSTEMS

Hari Krishna

A Master Thesis
in
The Department
of
Electrical Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Engineering at
Concordia University
Montreal, Quebec, Canada

December 1982

© H. Krishna 1982
ABSTRACT

A HIGH RATE CODING SCHEME FOR
BYTE-ORIENTED INFORMATION SYSTEMS

Hari Krishna

In this thesis, the mapping of non-binary codes defined over GF(2^m) into binary codes is studied and a binary coding scheme is derived that can be used to provide additional error-protection from random and burst errors for data consisting of bytes that have even parity. The emphasis here is on high rate codes.

A closed form decoding algorithm is given for the cases when two or three parity bytes are added to every block of k information bytes. An erasure is defined in the context of such systems. The decoding algorithm makes use of the parity bit that is present in every received byte.

Finally, the statistical performance of the coding scheme is analysed for the cases of interest on the non-binary and the binary symmetric channel models. For the non-binary symmetric channel, the Hamming weight distribution of the code is used to compute the probability of various post decoding events. The complete weight enumerator of the dual code and MacWilliams theorem are utilized for evaluating the performance on the binary symmetric channel.
To my parents
ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to Dr. Vijay K. Bhargava and Dr. Gérald E. Séguin for their constant encouragement, help and guidance throughout the course of this research work. I owe much of my understanding of algebraic coding theory to them.

I would also like to thank Miss Anita Claassen for her excellent typing of the manuscript.

Finally, I thank all my friends who have their office in the Electrical Engineering Graduate Research Laboratory for their cooperation and help from time to time.

This research was supported by the Natural Sciences and Engineering Research Council of Canada under grant G 0649.
TABLE OF CONTENTS

List of Figures vi
List of Important Symbols vii

CHAPTER 1. INTRODUCTION

1.1 Linear Block Codes 2
1.2 Cyclic Codes 2
1.3 Shortened Cyclic Codes 5
1.4 Plan of the Thesis 6

CHAPTER 2. A CODING SCHEME FOR BYTE-ORIENTED INFORMATION SYSTEMS 8

2.1 Mapping GF(2^m) Codes into Binary Codes 8
2.2 Encoding Scheme 9
2.3 Coding for d Equal to 3 11
2.4 Coding for d Equal to 4 14
2.5 Encoding 17

CHAPTER 3. THE DECODING ALGORITHM 21

3.1 A Closed Form Decoding Algorithm for d Equal to 3 22
3.2 A Closed Form Decoding Algorithm for d Equal to 4 25

CHAPTER 4. PERFORMANCE EVALUATION-I 38

4.1 Probabilistic Model of the Channel 39
4.2 On the Probability of Undetected Error, P_{ud} 40
4.3 Post Decoder Error Distribution and Symbol Error Rate for (27,25,3) Code 41
4.4 Post Decoder Error Distribution and Symbol Error Rate for (28,25,4) Code 44
4.5 Discussion of Results 46
CHAPTER 5. PERFORMANCE EVALUATION – II

5.1 Definitions & Notations 53
5.2 On the Probability of Post Decoder
Events for (216,175) Binary Code 55
5.3 MacWilliams' Theorem for Complete
Weight Enumerator 64
5.4 Further Analysis 66
5.5 A Note on the Performance of (224,175)
Binary Code 71
5.6 Discussion of Results 72

CHAPTER 6. CONCLUSIONS 80

BIBLIOGRAPHY 82

APPENDIX A Table of Elements of GF(2^7) 84

APPENDIX B Weight Distribution of (27,25) and
(28,25) RS Codes 86
# LIST OF FIGURES

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2.1</td>
<td>Encoder for $(n,k)$ Code</td>
<td>19</td>
</tr>
<tr>
<td>Fig. 2.2(a)</td>
<td>Encoder for $(27,25)$ Code</td>
<td>20</td>
</tr>
<tr>
<td>Fig. 2.2(b)</td>
<td>Encoder for $(28,25)$ Code</td>
<td>20</td>
</tr>
<tr>
<td>Fig. 3.1</td>
<td>Flowchart for Decoder of $(27,25)$ Code</td>
<td>33</td>
</tr>
<tr>
<td>Fig. 3.2</td>
<td>Flowchart for Decoder of $(28,25)$ Code</td>
<td>35</td>
</tr>
<tr>
<td>Fig. 4.1</td>
<td>Probability of Undetected Error Vs. Input Symbol Error Rate</td>
<td>47</td>
</tr>
<tr>
<td>Fig. 4.2</td>
<td>Probability of Correct Decoding Vs. Input Symbol Error Rate</td>
<td>48</td>
</tr>
<tr>
<td>Fig. 4.3</td>
<td>Probability of Incorrect Decoding Vs. Input Symbol Error Rate</td>
<td>49</td>
</tr>
<tr>
<td>Fig. 4.4</td>
<td>Probability of Decoding Failure Vs. Input Symbol Error Rate</td>
<td>50</td>
</tr>
<tr>
<td>Fig. 4.5</td>
<td>Post Decoder Symbol Error Rate Vs. Input Symbol Error Rate</td>
<td>51</td>
</tr>
<tr>
<td>Fig. 5.1</td>
<td>The Binary Symmetric Channel</td>
<td>53</td>
</tr>
<tr>
<td>Fig. 5.2</td>
<td>Probability of Undetected Error Vs. Input Bit Error Rate for $(216,175)$ Binary Code</td>
<td>75</td>
</tr>
<tr>
<td>Fig. 5.3</td>
<td>Probability of Correct Decoding Vs. Input Bit Error Rate for Binary Codes</td>
<td>76</td>
</tr>
<tr>
<td>Fig. 5.4</td>
<td>Probability of Incorrect Decoding Vs. Input Bit Error Rate for $(216,175)$ Binary Code</td>
<td>77</td>
</tr>
<tr>
<td>Fig. 5.5</td>
<td>Probability of Decoding Failure Vs. Input Bit Error Rate for $(216,175)$ Binary Code</td>
<td>78</td>
</tr>
<tr>
<td>Fig. 5.6</td>
<td>Post Decoder Symbol Error Rate Vs. Input Bit Error Rate for $(216,175)$ Binary Code</td>
<td>79</td>
</tr>
</tbody>
</table>
LIST OF IMPORTANT SYMBOLS

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n,k)</td>
<td>Linear code of length n and k information symbols.</td>
</tr>
<tr>
<td>q</td>
<td>Number of distinct input symbols.</td>
</tr>
<tr>
<td>g(X)</td>
<td>Generator polynomial of a cyclic code.</td>
</tr>
<tr>
<td>GF(p^m)</td>
<td>Galois Field of p^m elements.</td>
</tr>
<tr>
<td>RS code</td>
<td>Reed-Solomon code.</td>
</tr>
<tr>
<td>d</td>
<td>Minimum distance of a (n,k) code.</td>
</tr>
<tr>
<td>α</td>
<td>Primitive element of GF(p^m).</td>
</tr>
<tr>
<td>G</td>
<td>Generator matrix.</td>
</tr>
<tr>
<td>c(X)</td>
<td>Code polynomial.</td>
</tr>
<tr>
<td>e(X)</td>
<td>Error polynomial.</td>
</tr>
<tr>
<td>S_i</td>
<td>i-th syndrome.</td>
</tr>
<tr>
<td>P_{ud}</td>
<td>The probability of undetected error.</td>
</tr>
<tr>
<td>P_{CD}</td>
<td>The probability of correct decoding.</td>
</tr>
<tr>
<td>P_{ICD}</td>
<td>The probability of incorrect decoding.</td>
</tr>
<tr>
<td>P_{F}</td>
<td>The probability of decoding failure.</td>
</tr>
<tr>
<td>P_{SE}</td>
<td>Post decoder symbol error rate.</td>
</tr>
<tr>
<td>ε</td>
<td>Input symbol error rate.</td>
</tr>
<tr>
<td>P</td>
<td>Input bit error rate.</td>
</tr>
<tr>
<td>A(h)</td>
<td>Number of code words of Hamming weight h in a code.</td>
</tr>
<tr>
<td>a</td>
<td>Linear (n,k) code.</td>
</tr>
<tr>
<td>a^⊥</td>
<td>Dual of a linear (n,k) code.</td>
</tr>
<tr>
<td>x_β</td>
<td>A character of GF(q).</td>
</tr>
</tbody>
</table>
CHAPTER 1
Introduction

In this thesis, a coding scheme is presented for application to even parity byte-oriented information systems (for example, data in ASCII format, Teledon Broadcast System). In such systems, the information consists of \( k \) information bytes \( b_1, b_2, \ldots, b_k \), where each \( b_i \) consists of eight bits of even parity (although a similar analysis can be performed for odd parity). The constraints in encoding such information are:

1. The parity bit in each information byte cannot be altered, i.e., the encoding must be systematic.
2. The number of overall parity bits used must be a multiple of 8 and
3. The rate of the code should be high.

It should be emphasized that if the parity bits in the information bytes are allowed to be recomputed in a manner up to the system designer then the coding problem reduces to the standard coding problem and the designer can use any one of the available error correcting codes such as the BCH codes [1]. Since the parity bit is not to be changed, the overall performance of the coding scheme will depend on its use of these added parity bits in the error control process.

Readers not familiar with the theory of error correcting codes may find references [2], [3] and [4] useful. A comprehensive coverage of algebraic coding theory and of the relative interfaces between algebraic coding theory and surrounding areas is contained in reference [5]. Also the theory of rings and finite fields is not discussed here and the reader is referred to [3] for it.
In this chapter, linear block codes and other relevant topics are discussed briefly and the plan of the thesis is described.

1.1 Linear Block Codes

Let \( q \) denote the number of distinct symbols used on the channel. A block code is a set of \( M \) sequences of channel symbols of length \( n \). These \( q \)-ary \( n \)-tuples are called the code words of the code. The number of code words is taken to be a power of \( q \), i.e. \( M = q^k \).

The set of all \( n \)-tuples with entries chosen from the field of \( q \) elements is a vector space. A set of these vectors of length \( n \) is called a linear block code if and only if it is a subspace of the vector space of \( n \)-tuples. If the dimension of the subspace is \( k \), then such a code is called an \((n,k)\) code.

The Hamming distance between two vectors \( v_1 \) and \( v_2 \) is defined to be the number of positions in which the two vectors differ. The Hamming weight of a vector \( v \), denoted by \( w(v) \), is defined to be the number of non-zero components of \( v \). Thus Hamming distance between two vectors \( v_1 \) and \( v_2 \) is \( w(v_1 - v_2) \).

If \( c_1 \) and \( c_2 \) are both code words of a linear block code, then \( c_1 - c_2 \) must also be a code word, since the set of all code words is a vector space. Therefore, the distance between any two code words equals the weight of some other code word and the minimum distance \( d \) for a linear code equals the minimum weight of its non-zero vectors.

1.2 Cyclic Codes

A subspace \( V \) of \( n \)-tuples is called a cyclic subspace or a cyclic code if for each vector \( c = (c_0, c_1, \ldots, c_{n-1}) \) in \( V \), the vector
c' = (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) obtained by shifting the components of c cyclically one unit to the right is also in V. We will represent the components of a code vector as coefficients of a polynomial as follows

\[ c = (c_0, c_1, \ldots, c_{n-1}) \]
\[ c(X) = c_0 + c_1 X + \cdots + c_{n-1} X^{n-1} \] (1.2.1)

1.2.1 Generator Polynomial

Let \( g(X) = g_0 + g_1 X + \cdots + g_{r-1} X^{r-1} + X^r \) be a non-zero code polynomial of minimum degree in the \((n,k)\) cyclic code. It can be shown that \( r = n-k \) and a polynomial \( c(X) \) of degree \( n-1 \) or less is a code polynomial if and only if \( c(X) \) is a multiple of \( g(X) \).

The polynomial \( g(X) \) is called the generator polynomial of the cyclic code. Thus every code polynomial \( c(X) \) in an \((n,k)\) cyclic code can be expressed as

\[ c(X) = m(X) \cdot g(X) \]
\[ = (m_0 + m_1 X + \cdots + m_{k-1} X^{k-1}) \cdot g(X) \] (1.2.2)

where \( m(X) \) is the polynomial corresponding to the \( k \) information digits \((m_0, m_1, \ldots, m_{k-1})\).

1.2.2 Minimum Polynomial

Let \( \alpha \) be an arbitrary element of the Galois field \( GF(2^m) \).

The monic\(^1\) polynomial \( m(X) \) of smallest degree with binary coefficients such that \( m(\alpha) = 0 \), is called the minimum polynomial of \( \alpha \). The minimum polynomial of \( \alpha \) is irreducible.

\(^1\) A polynomial is called monic if the coefficient of the highest power of \( X \) is 1.
1.2.3 BCH Codes

Let \( \alpha \) be an element of \( \text{GF}(p^m) \). For any specified \( m_0 \) and \( d_0 \), the code generated by \( g(X) \) is a BCH code if and only if \( g(X) \) is the polynomial of lowest degree over \( \text{GF}(p) \) having \( \alpha, \alpha^2, \ldots, \alpha^{m_0 + d_0 - 2} \) as its roots. The length of the code is the least common multiple of the orders of the roots. The minimum distance of the code is at least \( d_0 \) and \( d_0 \) is called the designed distance \([3]\). The most important BCH codes are the binary codes obtained by letting \( \alpha \) be a primitive element of \( \text{GF}(2^m) \) and letting \( m_0 = 1 \) and \( d_0 = 2t_0 + 1 \). The generator polynomial of the \( t_0 \) error correcting code has \( \alpha, \alpha^2, \ldots, \alpha^{2t_0} \) as its roots and is given by

\[
g(X) = \text{LCM}(m_1(X), m_2(X), \ldots, m_{2t_0}(X))
\] (1.2.3)

However, every even power of \( \alpha \) is a root of the same minimum function as some previous odd power of \( \alpha \). Thus the generator polynomial of the code is

\[
g(X) = \text{LCM}(m_1(X), m_3(X), \ldots, m_{2t_0+1}(X))
\] (1.2.4)

As the degree of each minimum polynomial is \( m \) or less, the degree of \( g(X) \) is at most \( mt_0 \) and the code has at most \( mt_0 \) parity checks. Hence such a code has the following parameters.

Block length:

\[ n = 2^m - 1 \]

Number of parity check bits: \( n-k \leq mt_0 \) (1.2.5)

Minimum distance:

\[ d > 2t_0 + 1 \]

This code is capable of correcting any combination of \( t_0 \) or fewer errors in a block of \( n = 2^m - 1 \) bits.
1.2.4 Reed-Solomon Codes [6]

A Reed-Solomon code is a BCH code of length \( n = q - 1 \) over \( GF(q) \),
where \( q = p^m \) and \( p \) is a prime number. Of course, \( q \) is never equal to 2.
Thus the length of the code is the number of non-zero elements in the ground field.

The generator polynomial for such a code has \( \alpha, \alpha^2, \ldots, \alpha^{\delta - 1} \) as its roots for the minimum distance to be \( \delta \). Since the minimal polynomial
of \( \alpha^i \) is \( m^{(i)}(X) = (X - \alpha^i) \), a RS code of length \( q - 1 \) and designed distance \( \delta \) has generator polynomial

\[
g(X) = (X - \alpha)(X - \alpha^2) \cdots (X - \alpha^{\delta - 1}). \tag{1.2.6}
\]

As the degree of \( g(X) \) is \( \delta - 1 \), the RS code generated by \( g(X) \) has the following parameters:

Block length: \( n = q - 1 \) symbols

Number of parity symbols: \( n - k = \delta - 1 \) \tag{1.2.7}

Minimum distance: \( d = \delta \)

Since \( d = n - k + 1 \) for RS codes, these codes are called "maximum distance separable".

1.3 Shortened Cyclic Codes

In system design, if a code of suitable block length \( n \) or suitable dimension \( k \) cannot be found, it is natural to look for linear codes that, though actually not cyclic, share the mathematical structure and ease of implementation of cyclic codes. A technique for shortening a cyclic code is described in the following.

Given an \((n, k)\) linear code, it is always possible to form an \((n-i, k-i)\) linear code by making the \( i \) leading information symbols
identically 0 and omit them from all code vectors. This corresponds to omitting the first \( i \) rows and columns of the generator matrix or the first \( i \) columns from the parity check matrix. The resulting code is called a shortened code and in general, is not cyclic.

A shortened code has at least the same error-correcting capability as the code from which it is derived.

1.4 Plan Of The Thesis

The thesis is divided into six chapters and a very brief description of these chapters is as follows.

In chapter 2, the encoding scheme is described for the byte oriented information systems. The generator matrix is obtained in systematic form for the values of \( d \) equal to 3 and 4. The encoding procedure is analysed for system implementation.

In chapter 3, the decoding algorithm is discussed. A simplified closed form decoding algorithm is given by Séguin [7] for \( d \) equal to 3. It has been further extended to the case of \( d \) equal to 4. The decoding algorithm makes use of the parity bit that is present in every information and parity bytes.

In chapter 4, expressions are derived for the statistical performance of the coding scheme over the \( q \)-ary symmetric channel for \( d \) equal to 3 and 4. These expressions are evaluated using the computer for different values of channel symbol error rate.

In chapter 5, the performance of the code is analysed on a binary symmetric channel model for \( d \) equal to 3. The expressions for the probability of various post decoder events, are derived in terms of
the complete weight enumerator of the dual code which is generated on the computer.

Chapter 6 is the summary and conclusion.
CHAPTER 2

A Coding Scheme for Byte-Oriented Information Systems

This chapter describes a coding scheme that can be used very effectively to correct both random as well as burst errors. All the bytes in a code word computed on the basis of the information bytes, have even parity. The procedure for obtaining binary generator matrices in systematic form is explained and a method to implement the encoding is discussed briefly.

2.1 Mapping GF($2^m$) Codes into Binary Codes

We know that GF($p^m$) is a vector space of dimension $m$ over GF(p). Therefore, any set of $m$ linearly independent elements can be used as a basis for this vector space [1].

Let $\xi_1, \ldots, \xi_m$ be a basis for GF($p^m$) over GF(p). Then if $\beta = \sum_{i=1}^{m} b_i \xi_i$ is any element of GF($p^m$), $b_i \in$ GF(p), we map $\beta$ into $(b_1, b_2, \ldots, b_m)$. This mapping sends linear codes into linear codes (but cyclic codes may not map into cyclic codes).

Usually $1, \alpha, \ldots, \alpha^{m-1}$ is chosen to be the basis with $\alpha^i$ being represented as an $m$-tuple with only its $(i+1)$th or $(m-i)$th element as 1 and all other elements to be 0. With this basis, the elements of GF($p^m$) can be represented as $m$-tuples of elements from GF(p). Hence an $(n,k,d)$ RS code defined over GF($p^m$) becomes an $(n_b = mn, k_b = mk, d_b = d)$ code over GF(p). If $p = 2$, we get binary codes from this mapping.

Let $c = (c_0, c_1, \ldots, c_{n-1})$ belong to an $(n,k,d)$ RS code over GF($2^m$). If each of $c_i$ is replaced by a binary $m$-tuple according to the mapping given above and an overall parity check is added on each $m$-tuple, then the resulting binary code has the following parameters,
\[ n_b = (m + 1)(2^m - 1) \]

\[ k_b = mk \]

\[ d_b \geq 2(2^m - k) \]

for any \( k = 1, \ldots, 2^m - 2 \).

### 2.2 Encoding Scheme

For the type of information systems being considered here, the information is byte structured with each byte having an overall parity bit. Therefore, there are 128 (=2^7) different values that any information byte can take.

It can be readily observed that all the 8-tuples having even number of ones form a vector space of dimension 7. Again taking \( 1, \alpha, \ldots, \alpha^6 \) as the basis and representing them as,

\[
\begin{align*}
1 &= (1,0,0,0,0,0,0,1) \\
\alpha &= (1,0,0,0,0,0,1,0) \\
\alpha^2 &= (1,0,0,0,0,1,0,0) \\
\alpha^3 &= (1,0,0,0,1,0,0,0) \\
\alpha^4 &= (1,0,0,1,0,0,0,0) \\
\alpha^5 &= (1,0,1,0,0,0,0,0) \\
\alpha^6 &= (1,1,0,0,0,0,0,0)
\end{align*}
\]

it can be shown that all the even parity 8-tuples can be represented as elements of \( \text{GF}(2^7) \).

Hence let us define a RS code over \( \text{GF}(2^7) \) for such an information system. This code has the following parameters
Each symbol \( c_i \) in a code word obtained as a result of this encoding is written as \( c_i = b_0 a^0 + b_1 a + \ldots + b_6 a^6 \), \( b_j \in GF(2) \) and the corresponding even parity 8-tuple is obtained by replacing \( a^j \) by its 8-tuple representation defined by Equation (2.2.1).

Thus we have been able to introduce redundancy into each of the bytes that are transmitted. The parameters of the resulting binary code are:

\[
\begin{align*}
 n_b &= 8\cdot 127 \\
 k_b &= 7\cdot k \\
 d_b &\geq 2(128 - k). 
\end{align*}
\]

The code length and its dimensions can be shortened and the encoder can be put in the systematic form so as to match with the overall system requirements. A generator matrix can be obtained in the systematic form for the above RS code [3] and it is of the form

\[
G = [P, I_k]
\]

where \( I_k \) is the \( k \times k \) identity matrix and \( P \) is \( k \times (n-k) \) matrix. If \( m \) represents the information vector, then the encoded vector is of the form

\[
\begin{bmatrix} \text{mP} \\ m \end{bmatrix}
\]

The field \( GF(2^7) \) may be generated by the recursion

\[
a^7 = 1 + a^3
\]

where \( a \) is a primitive element of \( GF(2^7) \). Using this recursion and the basis defined by Equation (2.2.1), the complete table of \( GF(2^7) \) is
generated and is given in appendix A.

The complete encoding procedure can be described in the following three steps:

1. Represent each of the even parity information bytes as elements of GF(2^7).

2. Take k of these symbols and encode it using the RS code defined over GF(2^7) to get a code vector of length n.

3. Replace each one of the n symbols in the code word by its corresponding 8-tuple binary representation as explained above. Note that all the 8-tuples have even parity.

A binary generator matrix of size k_b x n_b can be obtained in systematic form for the above described encoding procedure. This is further illustrated for case of d equal to 3 and 4.

Since in many system applications, only codes of high rate can be considered, the two cases of d equal to 3 and 4 are of greater practical interest and, therefore, we will analyse these in detail.

2.3 Coding for d Equal to 3

The generator polynomial of a RS code defined over GF(2^7) and having a minimum distance d = 3, is

\[ g(X) = (X + \alpha)(X + \alpha^2) \]

\[ = X^2 + \alpha^{32}X + \alpha^3. \]

This polynomial generates a RS code of length 127. Let us assume that the code is to be shortened to k equal to 25. We do this in order to make the analysis more relevant for the signal formats described in [7] for Canadian Telidon System. Hence the shortened RS
code has the following parameters

\[ n' = 27 \]
\[ k = 25 \]
\[ d = 3 \]  \hspace{1cm} (2.3.1)

The generator matrix \( G \) in systematic form for this code is as follows.

\[
G = \begin{bmatrix}
3 & 32 & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\
35 & 105 & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \\
108 & 96 & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\
99 & 55 & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
58 & 115 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
118 & 105 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
108 & 97 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
100 & 21 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
24 & 15 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
18 & 13 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
16 & 8 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
1 & 72 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
75 & 115 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
116 & 46 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
49 & 99 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
102 & 54 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
57 & 74 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
77 & 93 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
96 & 62 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
65 & 29 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
32 & 58 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
61 & 107 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
110 & 48 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
51 & 79 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
82 & 49 & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\vdots & \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots \\
\end{bmatrix}
\]

\( (2.3.2) \)
After mapping the above code into the binary code according to the encoding procedure described in section (2.2), we get a binary code having the following parameters

\[ n_b = 216 \]
\[ k_b = 175 \]
\[ d_b \geq 6. \]  

(2.3.3)

In order for this mapping of the RS code into the binary code to be complete, we should obtain the corresponding generator matrix having only binary elements.

It can be shown that the multiplication between \( \alpha_1 \) and \( \alpha_2 \), \( \alpha_1, \alpha_2 \in \text{GF}(p^m) \) can be performed as a matrix multiplication of the form

\[ \bar{a}_1 \times B = \bar{a}_2 \]

where

- \( \bar{a}_1 \) represents \( \alpha_1 \) as \( m' \)-tuple of elements from \( \text{GF}(p) \), \( m' > m \),
- \( B \) is \( m' \times m' \) matrix of elements from \( \text{GF}(p) \) determined uniquely from \( \alpha_2 \) and
- \( \bar{a}_2 \) represents the product \( \alpha_1 \alpha_2 \) as an \( m' \)-tuple of elements from \( \text{GF}(p) \).

Example (2.3.1)

In our case

\[ p = 2, \ m = 7, \ m' = 8 \]

and for the basis defined by Equation (2.2.1), the multiplication by \( \alpha = (1 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0) \in \text{GF}(2^7) \) can be performed by taking \( B \) as
Generalizing the multiplication procedure, it can be shown that the binary matrix that corresponds to a multiplication by an element $a_i$ is given by

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

(2.3.4)

$$
\begin{bmatrix}
0 \\
b_{i+6} \\
b_{i+5} \\
b_{i+4} \\
b_{i+3} \\
b_{i+2} \\
b_{i+1} \\
b_i \\
\end{bmatrix}
$$

(2.3.5)

where each of $b_i, \ldots, b_{i+6}$ is a binary 8-tuple corresponding to $a_i, \ldots, a_{i+6}$ respectively and 0 is the all 0 8-tuple.

Hence the binary generator matrix can be obtained by replacing each element of the generator matrix $G$ given in Equation (2.3.2) by its corresponding $8 \times 8$ binary matrix as described above.

2.4 Coding for $d$ Equal to 4

The generator polynomial of the RS code for $d$ equal to 4 is given by
\[ g(X) = (X + \alpha)(X + \alpha^2)(X + \alpha^3) \]
\[ = X^3 + \alpha 104X^2 + \alpha 106X + \alpha^6. \]

This polynomial generates a RS code of length 127. As was explained in section (2.3), let the code be shortened to \( k = 25 \). Hence the shortened code has the following parameters

\[
\begin{align*}
n &= 28 \\
k &= 25 \\
d &= 4
\end{align*}
\]

(2.4.1)

The generator matrix \( G \) in systematic form for this code is as follows
After mapping the above code into a binary code according to the encoding procedure described in section (2.2), we get a binary code having the following parameters:

\[
\begin{align*}
\mathbf{n}_b &= 224 \\
\mathbf{k}_b &= 175 \\
\mathbf{d}_b &> 8
\end{align*}
\]
The binary generator matrix corresponding to the generator matrix given above can be calculated according to the procedure described in section (2.3).

2.5 Encoding

There are two methods for encoding linear cyclic codes - the serial shift register method and the parallel matrix method.

Let \( m(X) \) be a message polynomial with \( k \) symbols encoded into a code polynomial \( c(X) \) with \( n \) symbols. In the serial shift register method, encoding in systematic form is done by dividing \( X^{n-k} m(X) \) by \( g(X) \) and appending the remainder \( r(X) \) to \( X^{n-k} m(X) \). Thus

\[
  c(X) = r(X) + X^{n-k} m(X) = q(X) g(X)
\]

(2.5.1)

where \( q(X) \) is the quotient. It indicates that \( [r(X) + X^{n-k} m(X)] \) is a multiple of \( g(X) \), and, therefore, is a code polynomial generated by \( g(X) \).

The code word generated is given by

\[
  (r_0, r_1, \ldots, r_{n-k-1}, m_0, m_1, \ldots, m_{k-1})
\]

- parity check symbols
- message symbols

and the most significant symbol of the message, \( m_{k-1} \), is sent first.

Equation (2.5.1) can be implemented by a dividing circuit, which is \( (n-k) \)-stage shift register with feedback connections according to the generator polynomial. The feedback multipliers \( g_0, g_1, \ldots, g_{n-k-1} \) are coefficients of the generator polynomial

\[
  g(X) = (X - \alpha)(X - \alpha^2) \cdots (X - \alpha^{n-k}).
\]
An encoding circuit with an \((n-k)\)-stage shift register is shown in Figure (2.1). In our case, each \(r_i\) register stage is a 8-tuple shift register. The encoding is accomplished as follows. With gate turned on, \(k\) information symbols are shifted into the encoder and simultaneously sent into the communication channel. Then the gate is turned off and the contents of the shift register are shifted out to the channel.

Figure (2.2) shows the \((n-k)\)-stage shift register encoding circuit for the minimum distance \(d\) of the encoding scheme to be 3 and 4.

The parallel matrix method is more complex as compared to the serial method and is not described here. A binary generator matrix can be obtained for the coding scheme using the procedure given in section (2.3) and then encoding can be performed using this method. The reader is referred to [8] where the parallel matrix method is used to design and construct a \((75,50)\) forward error correcting codec.
Fig. 2.1. Encoder for \( (n,k) \) code: \( g_i \) is an element of \( \text{GF}(2^7) \) and \( r_i \) is an 8-tuple shift register stage.
FIG. 2.2(a). ENCODER FOR (27,25) CODE

FIG. 2.2(b). ENCODER FOR (28,25) CODE
CHAPTER 3

The Decoding Algorithm

For the coding scheme described in chapter 2, all the bytes of any given code vector have even parity. To make use of this inherent redundancy in the code vector bytes, the decoder first verifies the parity in the received bytes \( r_0, r_1, \ldots, r_{n-1} \). If any of them do not check out then there is a detectable error (though the value of the error is not known) in that position. Such a position is termed as an "erasure". The advantage of this becomes readily apparent by noting that a RS code with minimum distance \( d \) can correct \( t \) errors and \( s \) erasures provided

\[ 2t + s < d. \]

Hence the amount of redundancy required to correct an erasure is only half of that required to correct an error.

The Berlekamp-Massey decoding algorithm described in [5] can be used to correct the simultaneous occurrence of errors and erasures. The frequency domain decoding algorithm and the computational complexity associated with them has been discussed in [7] and it is shown that it is more promising to implement the transform technique for decoding erasures and errors of RS codes as compared to the time domain technique. The transform computations are independent of code rate and, therefore, the transform decoder is most efficient for low rate codes.

As the main emphasis in this thesis work is on high rate codes, a closed form decoding algorithm is described below for the minimum distance \( d \) of the code equal to 3 and 4. The decoding algorithm for \( d \) equal to 3 is given by Séguin [7] and here, it is extended to the case
of \( d \) equal to 4. This algorithm makes use of the parity bit present in every received byte.

3.1 A Closed Form Decoding Algorithm for \( d \) Equal to 3

For the encoding scheme given in section (2.3), the minimum distance of the code is 3 and, therefore, it can correct \( t \) errors and \( s \) erasures if

\[
2t + s < 3 \quad (3.1.1)
\]

It is clear from Equation (3.1.1) that the received vector is correctly decoded to the transmitted code word, iff
(a) \( s \leq 2, t = 0 \), i.e. a maximum of two erasures and no error takes place.
(b) \( t \leq 1, s = 0 \), i.e. a maximum of one error and no erasure takes place.

Thus, the following decoding procedure is viable for such an encoding scheme.

(i) Two Detectable Errors in the Received Bytes

Let the error polynomial be

\[
e(X) = e_i X^i + e_j X^j \quad (3.1.2)
\]

where \( e_i, e_j \in \text{GF}(2^7) \) and \( 1 \leq i < j \) and let \( r(X) \) be the received polynomial, i.e.

\[
r(X) = c(X) + e(X) \quad (3.1.3)
\]

where \( c(X) \) is the transmitted code vector. Then the two syndromes \( S_1 \) and \( S_2 \) are given by

\[
S_1 = r(a) = e_i a^i + e_j a^j
\]
\[
S_2 = r(a^2) = e_i a^{2i} + e_j a^{2j} \quad (3.1.4)
\]

and the values of \( i \) and \( j \) are known from the parity bits associated
with the bytes in positions $i+1$ and $j+1$. We then have

$$
(e_i, e_j) \begin{bmatrix} \alpha^i & \alpha^{2i} \\ \alpha^j & \alpha^{2j} \end{bmatrix} = (S_1, S_2)
$$

(3.1.5)

The latter matrix has an inverse which is

$$
\frac{1}{\Delta} \begin{bmatrix} \alpha^{2j} & \alpha^{2i} \\ \alpha^j & \alpha^i \end{bmatrix}
$$

(3.1.6)

where $\Delta = \alpha^{i+2j} + \alpha^{2i+j}$.

Multiplying both sides of Equation (3.1.5) by (3.1.6), we obtain

$$
e_i = \frac{S_1\alpha^{2j} + S_2\alpha^j}{\Delta}
$$

(3.1.7)

$$
e_j = \frac{S_1\alpha^{2i} + S_2\alpha^i}{\Delta}
$$

(3.1.8)

Hence the errors in positions $i$ and $j$ are determined by Equations (3.1.7) and (3.1.8) respectively.

(ii) A Single Detectable Error in the Received Bytes

Suppose there is a detectable error in position $i$, then the error polynomial is given by

$$
e(X) = e_i X^i
$$

where $e_i \in GF(2^7)$ and the syndromes $S_1$ and $S_2$ are

$$
S_1 = r(\alpha) = e_i \alpha^i
$$

(3.1.9)

$$
S_2 = r(\alpha^2) = e_i \alpha^{2i}
$$

The value of $i$ is known. Thus,
\[ e_i = a^{-i}S_1 \]  
(3.1.10)

Also, it can be observed that
\[ S_2 = aS_1^i \]  
(3.1.11)

The Equation (3.1.10) can be used to calculate the value of the error at position \( i \).

(iii) A Single Byte in Error

If there is a single byte in error, then the error polynomial is given by
\[ e(x) = e_i x^i \]

where \( e_i \in GF(2^7) \) and \( i \) is not known.

The two syndromes \( S_1 \) and \( S_2 \) are calculated as
\[ S_1 = r(a) = e_i a^i \]  
(3.1.12)
\[ S_2 = r(a^2) = e_i a^{2i} \]

Solving Equation (3.1.12) for the value of \( e_i \) and \( i \), we get
\[ e_i = S_2 S_2^{-1} \]  
(3.1.13)
\[ a^i = S_2 S_1^{-1} \]

A look-up table of \( (i, a^i) \) can be used to determine \( i \).

Hence, the decoding algorithm for the coding scheme presented in section (2.3) can be described as follows

Step I: Check the parity of each of the received bytes.

- If they all check, go to II
- If exactly one does not check, go to III
If exactly two do not check, go to IV
If more than two do not check, declare a decoding failure.
Proceed to next frame.

Step II : Represent all the bytes as elements of \( \text{GF}(2^7) \) and compute the syndromes \( S_1 = r(a) \) and \( S_2 = r(a^2) \).
If both are 0, assume \( r(X) \) is error free.
If both are nonzero, assume a single symbol in error and decode it using the procedure (iii) described above.
If exactly one is 0, declare a decoding failure. Proceed to next frame.

Step III: Compute the syndromes. Check for Equation (3.1.11). If it is satisfied, decode it using procedure (ii) given above.
If Equation (3.1.11) is not satisfied, declare a decoding failure. Proceed to next frame.

Step IV: Assume exactly two symbols in error and correct them using procedure (i) described above. Proceed to next frame.

The above decoding algorithm ensures that a decoded code word has bytes that have even parity only. The complete decoding algorithm is given by the flowchart in Figure (3.1).

3.2 A Closed Form Decoding Algorithm for \( d \) Equal to 4

The encoding scheme and the generator matrix for \( d \) equal to 4 are given in section (3.4). It can correct \( t \) errors and \( s \) erasures if

\[
2t + s < -4
\]  
(3.2.1)
Thus a received vector is correctly decoded to the transmitted code word if one of the following combinations of errors and erasures takes place:

(a) No erasure, no error \((2t+s=0)\)
(b) One erasure, no error \((2t+s=1)\)
(c) Two erasures, no error \((2t+s=2)\)
(d) Three erasures, no error \((2t+s=3)\)
(e) One error, no erasure \((2t+s=2)\)
(f) One error, one erasure \((2t+s=3)\).

A simplified decoding procedure described below is viable for such a coding scheme.

(i) A Single Detectable Error in the Received Bytes

Let there be a detectable error in position \(i\), then the error polynomial is

\[ e(X) = e_i X^i \]

where \(e_i \in GF(2^7)\). The syndromes \(S_1\) and \(S_2\) and \(S_3\) are given by

\[
\begin{align*}
S_1 &= r(\alpha) = e_i \alpha^i \\
S_2 &= r(\alpha^2) = e_i \alpha^{2i} \\
S_3 &= r(\alpha^3) = e_i \alpha^{3i}
\end{align*}
\]  

(3.2.2)

Since the value of \(i\) is known, solving Equation (3.2.2) for \(e_i\), we get

\[ e_i = \alpha^{-i} S_1 \]  

(3.2.3)

Also we have to perform a check that there is no error in the received bytes. This can be done by noting that
\[ S_2 = e_i a^{2i} = a^{-i} S_1 a^{2i} = S_1 a^i \] (3.2.4)

and

\[ S_3 = e_i a^{3i} = a^{-i} S_1 a^{3i} = S_1 a^{2i} \] (3.2.5)

Combining Equations (3.2.4) and (3.2.5), we get

\[ S_3 S_1 = S_2^2 \] (3.2.6)

Hence if there is only one detectable error, then

\[ e_i = a^{-i} S_1 \]

and

\[ S_3 S_1 = S_2^2 \]

(ii) Two Detectable Errors in the Received Bytes

Let there be detectable errors in positions \( i \) and \( j \). The procedure for finding the error values at these positions is given in section (3.1). Additionally, we have to check if there are any other errors in the received bytes. It can be done by observing that

\[ S_3 = e_i a^{3i} + e_j a^{3j} \] (3.2.7)

Substituting for \( e_i \) and \( e_j \) from Equations (3.1.7) and (3.1.8) respectively, we get

\[ S_3 = a^{(i+j)} S_1 + (a^i + a^j) S_2 \]

Thus if there are only two detectable errors, then

\[ e_i = \frac{S_1 a^{2j} + S_2 a^j}{\Delta} \]
\[ e_j = \frac{S_1 a^{2i} + S_2 a^j}{\Delta} \]  
(3.2.8)

where \( \Delta = a^{i+2j} + a^{j+2i} \),

and \( S_3 = a^{i+j} S_1 + (a^i + a^j) S_2 \)

(iii) Three Detectable Errors in the Received Bytes

If there are detectable errors in positions \( i, j \) and \( k \), then the error polynomial is

\[ e(x) = e_i x^i + e_j x^j + e_k x^k \]

where \( e_i, e_j, e_k \in GF(2^7) \) and \( 1 \leq i < j < k \).

The syndromes \( S_1, S_2 \) and \( S_3 \) calculated on the basis of the received vector are

\[ S_1 = e_i a^i + e_j a^j + e_k a^k \]
\[ S_2 = e_i a^{2i} + e_j a^{2j} + e_k a^{2k} \]  
(3.2.9)
\[ S_3 = e_i a^{3i} + e_j a^{3j} + e_k a^{3k} \]

Equation (3.2.9) can be rewritten in the matrix form as

\[
\begin{bmatrix}
  a^i & a^j & a^k \\
  a^{2i} & a^{2j} & a^{2k} \\
  a^{3i} & a^{3j} & a^{3k}
\end{bmatrix}
\begin{bmatrix}
  e_i \\
  e_j \\
  e_k
\end{bmatrix} =
\begin{bmatrix}
  S_1 \\
  S_2 \\
  S_3
\end{bmatrix}
\]
(3.2.10)

or

\[ Ae = S. \]

The matrix \( A \) has an inverse which is given by
\[ A^{-1} = \frac{1}{\Delta} \begin{bmatrix}
\alpha^{2j+3k} & \alpha^{3j+2k} & \alpha^{j+3k} & \alpha^{3j+k} & \alpha^{j+2k} \\
\alpha^{2i+3k} & \alpha^{3i+2k} & \alpha^{i+3k} & \alpha^{3i+k} & \alpha^{i+2k} \\
\alpha^{2i+3j} & \alpha^{3i+2j} & \alpha^{i+3j} & \alpha^{3i+j} & \alpha^{i+2j}
\end{bmatrix} \]

where

\[ \Delta = \alpha^{i+j+k} \left[ \alpha^{2i}(\alpha^j + \alpha^k) + \alpha^{2j}(\alpha^i + \alpha^k) + \alpha^{2k}(\alpha^i + \alpha^j) \right]. \]

Multiply both sides of equation (3.2.10) by \( A^{-1} \), we get:

\[ e_i = \frac{S_1(\alpha^{2j+3k} + \alpha^{3j+2k}) + S_2(\alpha^{j+3k} + \alpha^{3j+k}) + S_3(\alpha^{j+2k} + \alpha^{2j+k})}{\Delta} \]

\[ e_j = \frac{S_1(\alpha^{2i+3k} + \alpha^{3i+2k}) + S_2(\alpha^{i+3k} + \alpha^{3i+k}) + S_3(\alpha^{i+2k} + \alpha^{2i+k})}{\Delta} \]

\[ e_k = \frac{S_1(\alpha^{2i+3j} + \alpha^{3i+2j}) + S_2(\alpha^{i+3j} + \alpha^{3i+j}) + S_3(\alpha^{i+2j} + \alpha^{2i+j})}{\Delta} \]

Hence the erasure values are determined from the equation (3.2.12).

\((\text{i})\) One Error

The procedure for finding the error magnitude and its position is given in section (3.1). Also, we have to check if there is only one error in the received bytes. This can be done by observing that

\[ S_3 = r(\alpha^3) = e_i \alpha^{3i} \]

from which it follows that

\[ S_1S_3 = S_2^2 \]
(v) One Detectable and One Undetectable Error in the Received Bytes

Let there be a detectable error in position $i$ and an undetectable error in position $j$. The error polynomial, therefore, is given by

$$e(X) = e_i x^i + e_j x^j$$

where $e_i, e_j \in GF(2^7)$ and $i$ is known.

The syndromes $S_1, S_2$, and $S_3$ are then calculated as

$$S_1 = r(\alpha) = e_i \alpha^i + e_j \alpha^j$$
$$S_2 = r(\alpha^2) = e_i \alpha^{2j} + e_j \alpha^{2j} \tag{3.2.14}$$
$$S_3 = r(\alpha^3) = e_i \alpha^{3j} + e_j \alpha^{3j}$$

The value of $j$ is calculated from the above equation and is given by

$$\alpha^j = (S_3 + \alpha S_2)(S_2 + \alpha^i S_1)^{-1} \tag{3.2.15}$$

Once the location of error $j$ is determined, the magnitudes $e_i$ and $e_j$ can be calculated as in part (ii).

Hence, the decoding algorithm for the coding scheme described in chapter 2 for minimum distance $d$ equal to 4 can be stated as follows.

**Step I.** Check the parity of each of the received bytes.

- If all parity bits check, go to II.
- If exactly one does not check, go to III.
- If exactly two do not check, go to IV.
- If exactly three do not check, go to V.
If more than three do not check, declare a decoding failure. Proceed to next frame.

Step II

Compute the syndromes

\[ S_1 = r(\alpha), \quad S_2 = r(\alpha^2) \quad \text{and} \quad S_3 = r(\alpha^3). \]

If all three of \( S_1 \), \( S_2 \) and \( S_3 \) are zero, declare \( r(X) \) as error free.

If all of \( S_1 \), \( S_2 \) and \( S_3 \) are nonzero, assume a single error and decode it using the procedure (iv) given above. Also check if \( S_3S_1 = S_2^2 \). If not, declare a decoding failure.

If some of \( S_1 \), \( S_2 \) and \( S_3 \) are zero, declare a decoding failure.

Go to step VI.

Step III

Check if \( S_3S_1 = S_2^2 \). If yes, then go to III(a), else go to III(b).

III(a)

Assume a single detectable error and correct it using procedure (i) given above.

Go to step VI.

III(b)

Assume that an undetectable and a detectable error have occurred and decode it using procedure (v).

Go to step VI.

Step IV

Check if \( S_3 = \alpha^i + \beta S_1 + (\alpha^i + \alpha^j)S_2 \).

If yes, then assume that exactly two detectable errors have occurred and use procedure (ii) to decode them.

Go to step VI.

If not, declare a decoding failure.

Go to step VI.
Step V  Assume exactly three bytes in error and correct them using procedure (iii) given above.  
Go to step VI.

Step VI  Go to next frame.

Again, this decoding algorithm ensures that a decoded code word has bytes that have even parity only. The complete decoding algorithm is given by the flowchart in Figure (3.2).
FIG. 3.1. FLOWCHART FOR DECODER OF (27,25) CODE
FIG. 3.1. CONTINUED
FIG. 3.2. FLOWCHART FOR DECODER OF (28,25) CODE
FIG. 3.2. CONTINUED
FIG. 3.2. CONTINUED
CHAPTER 4

Performance Evaluation-I

In the selection of error-control coding technique, alternative coding schemes are compared on the basis of various probabilistic measures of performance and system configuration.

One basis of comparison can be the probability of undetected error $P_{ud}$, if the code is used for error detection only. The receiver in such a system, makes no attempt to correct errors but just checks if the received vector is a code word or not. However, errors may occur in a way that one transmitted code word is received as another code word and the probability of such an event is called the probability of undetectable error. This probability is calculated using the weight distribution of the code for the case of $d$ equal to 3 and 4.

Another basis for comparison is the probability of correct decoding $P_{CD}$, a quantity that can be calculated when the decoding algorithm is known and a memoryless channel can be assumed. Also, the probability of incorrect decoding $P_{ICD}$, the probability of decoding failure $P_F$ and the post decoder symbol error rate $P_{SE}$, can be useful in the evaluation of a coding scheme. Expressions for $P_{ICD}$, $P_{SE}$ are available in the literature for q-ary block codes with known weight distributions. The reader is referred to [9], [10] and [11] for these general expressions. Here, modified expressions are presented for the coding scheme described in chapter 2 and the decoding algorithm given in chapter 3 for the case of $d$ equal to 3 and 4. Plots of $P_{CD}$, $P_{ICD}$, $P_F$ and $P_{SE}$ versus the input symbol error rate $e$, are given for the two cases.
In the following performance analysis of the coding scheme, it has been assumed that the all zero code word is transmitted. However, the same analysis holds for the transmission of an arbitrary code word since the coding scheme being analysed is linear.

In this chapter, the performance of the coding scheme is analysed on a non-binary symmetric memoryless channel. The performance of the coding scheme for \( d \) equal to 3 is evaluated on a binary symmetric channel in chapter 5 using the complete enumerator of the dual code and MacWilliams theorem for complete weight enumerators.

4.1 Probabilistic Model of the Channel

The randomness associated with the transmission process has to be defined in order to be able to compute the probability of the various events of interest. For the code described in chapter 2, all the even parity 8-tuples were represented as elements of \( GF(2^7) \). However, in general, the set of all possible binary 8-tuples forms a vector space of dimension 8 and these 8-tuples can be used to represent elements of \( GF(2^8) \). Therefore, we will assume that any symbol that is transmitted has probability \((1-\varepsilon)\) of being received correctly and a probability of \(\varepsilon/(q' - 1)\) of being transformed into each of the \((q' - 1)\) other symbols, where \( q' = 2^8 = 256 \). This is based on the assumption that a received symbol can have either even or odd parity. Note that there are \( q'/2 = 128 \) elements of \( GF(2^8) \) that have odd number of ones in their binary 8-tuple representation and if any one of these symbols is received, it is termed as an erasure.

---

\(^1\) To be defined later
We also assume that successive symbols incur errors independently. Hence the probability that the received word differs from the transmitted word in exactly \( i \) positions is given by

\[
\binom{n}{i} (q' - 1)^i (1 - \varepsilon)^{n-i} = \binom{n}{i} \varepsilon^i (1 - \varepsilon)^{n-i}
\]

Note that if \( \varepsilon = (q' - 1)/q' \), each of the \( q' \) symbols from the alphabet occurs at the receiver with equal probability. Therefore, we consider the case when \( \varepsilon \leq (q' - 1)/q' \). One example of such a channel model is \( q \) - ary FSK modulation in additive white gaussian noise. Hence the probability that an error pattern having exactly \( i \) non-zero symbols will occur at the receiver is

\[
P(i) = \binom{n}{i} (1 - \varepsilon)^{n-i} \tag{4.1.1}
\]

4.2 On the Probability of Undetected Error \((P_{ud})\)

An error pattern will be accepted as a code word and lead to an undetected error if and only if it is the same as a non-zero code word \([12]\).

Let \( A(h) \) denote the number of code words having exactly \( h \) non-zero symbols. Then the probability of undetected error \( P_{ud} \) is given by

\[
P_{ud} = \sum_{h=1}^{n} A(h) P(h) \tag{4.2.1}
\]

where \( P(i) \) is defined by Equation (4.1.1).

The coding scheme described in chapter 2 is essentially a RS code defined over \( GF(2^7) \) and for such a code, the number of code words having exactly \( h \) non-zero symbols is given by
\[ A(h) = \binom{n}{h} (q-1)^{d-h} \sum_{i=0}^{h-d} (-1)^i \binom{h-1}{i} q^{h-d-i} \] (4.2.2)

for \( d \leq h \leq n \) and \( q = 2^7 = 128 \).

Thus for a given block length \( n \) and a minimum distance \( d \), the weight distribution \( A(h) \) can be calculated using Equation (4.2.2) and then \( P_{ud} \) is computed using Equation (4.2.1).

The above expressions have been evaluated for dimension of the code \( k \) equal to 25 and minimum distance \( d \) equal to 3 and 4. The weight distribution of (27,25,3) and (28,25,4) RS codes is given in appendix B. Plots of \( P_{ud} \) versus \( \epsilon \) are given in Figure 4.1 for the above codes.

4.3 Post Decoder Error Distribution and Symbol Error Rate for (27,25,3) Code

4.3.1 The Probability of Correct Decoding, \( P_{CD} \)

It was shown in section (3.1) that this code decodes correctly the patterns that correspond to the following events

1. No erasure, no error
2. One erasure, no error
3. Two erasures, no error
4. One error, no erasures.

The probability of each of these events is calculated as below

\[ P_{\text{event 1}} = P(0) \]

\[ P_{\text{event 2}} = \binom{n}{1} \left( \frac{q}{2} \right) P(1) \]
\[ P_{\text{event } 3} = \binom{n}{2} \left( \frac{q'}{2} \right)^2 P(2) \]

\[ P_{\text{event } 4} = \binom{n}{1} \left( \frac{q'}{2} - 1 \right) P(1) \]

Hence the probability of correct decoding \( P_{\text{CD}} \) is

\[ P_{\text{CD}} = P(0) + \binom{n}{1} \left( \frac{q'}{2} \right) P(1) + \binom{n}{2} \left( \frac{q'}{2} \right)^2 P(2) + \binom{n}{1} \left( \frac{q'}{2} - 1 \right) P(1) \]  \hspace{1cm} (4.3.1)

where \( n = 27, q' = 256 \).

4.3.2 The Probability of Incorrect Decoding, \( P_{\text{ICD}} \)

An incorrect decoding takes place if the received word is decoded to a code word other than the all 0 code word. It occurs if one of the four events described in (4.3.1) takes place with respect to a non-zero code word. Thus, if \( P_{\text{ICD}}(h) \) is the probability of incorrect decoding to a code word of weight \( h \), the probability of incorrect decoding \( P_{\text{ICD}} \) is

\[ P_{\text{ICD}} = \sum_{h=d}^{n} P_{\text{ICD}}(h) \]  \hspace{1cm} (4.3.2)

For a code word of weight \( h \), the probability of various events described in (4.3.1) is given by

\[ P_{\text{event } 1} = P(h) \]

\[ P_{\text{event } 2} = \binom{n}{1} \left( \frac{q'}{2} \right) P(h) + \binom{n-h}{1} \left( \frac{q'}{2} \right) P(h+1) \]
\[
P_{\text{event 3}} = \binom{h}{2} (q'/2)^2 p(h) + \binom{n-h}{2} (q'/2)^2 p(h+2)
+ \binom{h}{1} \binom{n-h}{1} (q'/2)^2 p(h+1)
\]

\[
P_{\text{event 4}} = \binom{h}{1} (q'/2-2)p(h) + \binom{h}{1} p(h-1) + \binom{n-h}{1} (q'/2-1)p(h+1)
\]

and

\[
\sum P_{\text{ICD}}(h) = A(h)(P_{\text{event 1}} + P_{\text{event 2}} + P_{\text{event 3}} + P_{\text{event 4}})
\]  
(4.3.3)

where \(A(h)\) denotes the number of code words having exactly \(h\) non-zero symbols and is calculated using Equation (4.2.2). \(P_{\text{ICD}}(h)\) is calculated for the values of \(h\) going from \(d\) to \(n\) and these values are substituted in Equation (4.3.2) to calculate \(P_{\text{ICD}}\). Also if \(P_F\) is the probability of decoding failure, then

\[
P_{\text{CD}} + P_{\text{ICD}} + P_F = 1
\]

and, therefore,

\[
P_F = 1 - P_{\text{CD}} - P_{\text{ICD}}
\]  
(4.3.4)

4.3.3 Post Decoding Symbol Error Rate, \(P_{SE}\)

The post decoding symbol error rate \(P_{SE}\) is defined as the expected number of errors in a code word following decoding. Hence

\[
P_{SE} = \frac{1}{n} \sum_{h=d}^{n} h P_{\text{ICD}}(h)
\]  
(4.3.5)

and can be calculated easily once \(P_{\text{ICD}}(h), d \leq h \leq n\) is known.
Plots of $P_{CD}$, $P_{ICD}$, $P_F$ and $P_{SE}$ versus $\varepsilon$ are given in Figures (4.2), (4.3), (4.4) and (4.5) respectively.

4.4 Post Decoder Error Distribution and Symbol Error Rate for $(28,25,4)$ Code

4.4.1 The Probability of Correct Decoding, $P_{CD}$

This code can be decoded correctly if the received patterns correspond to one of the following events

1. No erasure, no error
2. One erasure, no error
3. Two erasures, no error
4. Three erasures, no error
5. One error, no erasure
6. One error, one erasure.

The expression for the probability of events (1), (2), (3) and (5) is given in section (4.3.1) and

$$P_{\text{event 4}} = \binom{n}{3} (q'/2)^3 P(3)$$

$$P_{\text{event 6}} = 2 \binom{n}{2} (q'/2)(q'/2-1) P(2).$$

Hence

$$P_{CD} = P(0) + \binom{n}{1}(q'/2)P(1) + \binom{n}{2}(q'/2)^2 P(2) + \binom{n}{3} (q'/2)^3 P(3)$$

$$+ \binom{n}{1}(q'/2-1)P(1) + 2 \binom{n}{2} (q'/2)(q'/2-1)P(2) \quad (4.4.1)$$

where $n = 28$, $q' = 256$. 
4.4.2 The Probability of Incorrect Decoding

An incorrect decoding takes place if one of the 6 events given in section (4.4.1) takes place with respect to a non-zero code word. The events (1), (2), (3) and (5) correspond to events (1), (2), (3) and (4) of section (4.3.1) respectively and the expressions for the probability are given in section (4.3.2). For a code word of weight \( h \), the probability of events (4) and (6) is given by

\[
P_{\text{event 4}} = \binom{h}{3} (q'/2)^3 p(h) + \binom{n-h}{3} (q'/2)^3 p(h+3)
\]

\[
+ \binom{h}{2} (n-h) (q'/2)^3 p(h+1) + \binom{h}{1} (n-h) (q'/2)^3 p(h+2)
\]

\[
P_{\text{event 6}} = 2 \binom{h}{2} (q'/2) (q'/2-2) p(h) + 2 \binom{h}{1} (q'/2) p(h-1)
\]

\[
+ 2 \binom{n-h}{2} (q'/2) (q'/2-1) p(h+2) + \binom{h}{1} (n-h) (q'/2-2) (q'/2) p(h+1)
\]

\[
+ \binom{h}{1} (n-h) (q'/2) p(h) + \binom{n-h}{1} (q'/2) (q'/2-1) p(h+1)
\]

and

\[
P_{\text{ICD}}(h) = A(h) \left( \sum_{i=1}^{6} P_{\text{event i}} \right)
\] (4.4.2)

Hence \( P_{\text{ICD}} \) can be calculated using Equation (4.3.2). Also Equations (4.3.4) and (4.3.5) can be used to calculate \( P_F \) and \( P_{SE} \) respectively. Plots of \( P_{CD} \), \( P_{ICD} \), \( P_F \) and \( P_{SE} \) versus \( \varepsilon \) are given in Figures (4.2), (4.3), (4.4) and (4.5) respectively.
4.5 Discussion of Results

As it is clear from Figure (4.1), these codes can be used effectively for error control in systems, where the codes are employed for the purpose of error detection only. For noisy channels ($\varepsilon \sim 10^{-2}$), the $(27,25,3)$ code has $P_{ud}$ of the order of $10^{-8}$ while $(28,25,4)$ code has $P_{ud}$ of the order of $10^{-11}$. However, the receiver has to check for one more syndrome to be zero in latter code, which may not be a very high price to pay for the lower value of $P_{ud}$ in many systems where high reliability on the received information is required.

One interesting result obtained from the analysis performed in this chapter is that the probability of incorrect decoding is much lower as compared to the probability of decoding failure. Thus, most of the time the decoder either decodes the received message block correctly or it declares a decoding failure. It can particularly be helpful in situations where failure to decode can be tolerated but the penalty that one pays for decoding incorrectly is high.

For $\varepsilon$ of the order of $10^{-2}$, the post decoder symbol error rate, $P_{SE}$ is of the order of $10^{-5}$ for the $(28,25,4)$ code while it is of the order of $10^{-4}$ for the $(27,25,3)$ code. Thus the additional complexity of the decoding algorithm in case of $(28,25,4)$ code leads to significant reduction in the post decoder symbol error rate. These values could be quite acceptable for most digital communication systems. If still lower value of $P_{SE}$ is required, then a code of higher redundancy may be designed. But it must be pointed out that though the additional complexity in the encoder would be marginal, the decoding algorithm will no longer be in closed form and a more general, Berlekamp-Massey algorithm would be needed to perform the decoding.
FIG. 4.1. PROBABILITY OF UNDETECTED ERROR VS. INPUT SYMBOL ERROR RATE
Fig. 4.2: PROBABILITY OF CORRECT DECODING Vs.
INPUT SYMBOL ERROR RATE
Fig. 4.3. Probability of Incorrect Decoding vs. Input Symbol Error Rate
FIG. 4.4. PROBABILITY OF DECODING FAILURE VS. INPUT SYMBOL ERROR RATE
FIG. 4.5. POST DECODER SYMBOL ERROR RATES VS. INPUT SYMBOL ERROR RATE.
CHAPTER 5

Performance Evaluation-II

In chapter 4, the statistical performance of the coding scheme was analysed on a q-ary symmetric memoryless channel. The RS code defined on $GF(2^7)$ is mapped into a binary code as described in chapter 2, and therefore, it is interesting to evaluate the performance of the binary code obtained as a result of this mapping, on a binary channel. Though most communication channels are not accurately represented by the binary symmetric channel (BSC), shown in Figure (5.1), it has been studied extensively. For the binary symmetric channel, the probability is $Q$ that the same symbol will be received as transmitted. It is assumed that $Q > P$ and that each symbol is independent of all others. The example of such a channel model can be PSK, FSK and QPSK modulation in additive white gaussian noise with hard decision decoding.

In this chapter, expressions for $P_{CD}$, $P_{ICD}$, $P_{SE}$ are derived for the coding scheme and the decoding algorithm described in earlier chapters. However, since the analysis is very complicated, it has been carried out for $d$ equal to 3 only. For the case of $d$ equal to 4, only the expression for $P_{CD}$ is presented and evaluated. It will be seen shortly that the analysis of a code can be based on the structure of the dual code. The dual of a linear code with generator matrix $G$ is defined as the linear code whose parity check matrix is the transpose of $G$.

As was also stated earlier, we evaluate the performance on the assumption that the all zero code vector is transmitted. But since the code is linear, the same analysis holds for the transmission of any arbitrary code word.
Based on the assumption that the channel is a binary symmetric memoryless channel, the probability that an all zero 8-tuple is received as an 8-tuple having exactly \( i \) ones is \( p^i q^{8-i} \). Thus the probability that one 8-tuple gets converted to another 8-tuple when transmitted over this channel is \( p^i q^{8-i} \), where \( i \) is the number of positions the two 8-tuples differ. Hence the model of binary symmetric channel does not extend to a q-ary symmetric channel model.

5.1 Definitions, Notations

Let \( GF(q) \) be a Galois field of \( q \) elements, \( q = 2^m \). \( GF^n(q) \) is the set of all possible row vectors of length \( n \), in which each coordinate is an element of \( GF(q) \). Addition of two vectors is defined coordinate by coordinate, under the rules prevailing in \( GF(q) \). \( GF^n(q) \) is a vector space of dimension \( n \) over \( GF(q) \). Choose a basis consisting of \( n \) vectors

\[
\begin{align*}
    e_1 &= (1 0 0 \ldots 0) \\
    e_2 &= (0 1 0 \ldots 0) \\
        &\vdots \\
    e_n &= (0 0 0 \ldots 1)
\end{align*}
\]
An element \( u \) of \( \text{GF}^n(q) \) can be expressed uniquely as
\[
  u = \sum_{i=1}^{n} u_i e_i, \quad u_i \in \text{GF}(q) \quad (5.1.1)
\]

We can write \( u = (u_1, u_2, \ldots, u_n) \).

The Hamming weight of \( u \) is defined as the number of non-zero coordinates in \( u \).

5.1.1 Complete Weight Enumerator

Let the elements of \( \text{GF}(q) \) be denoted by \( \omega_0, \omega_1, \ldots, \omega_{q-1} \) in some fixed order. Complete weight enumerator classifies code words \( c \) in \( \text{GF}^n(q) \) according to the number of times each field element \( \omega_i \) appears in \( c \).

The composition of \( c = (c_0, c_1, \ldots, c_{n-1}) \) denoted by \( \text{comp}(c) \) is \( (s_0, s_1, \ldots, s_{q-1}) \), where \( s_i = s_i(c) \) is the number of components \( c_j \) equal to \( \omega_i \). Thus
\[
  \sum_{i=0}^{q-1} s_i = n.
\]

Let \( \mathcal{A} \) be a linear code over \( \text{GF}(q) \) and let \( A(t) \) be the number of code words \( c \in \mathcal{A} \) with \( \text{comp}(c) = t = (t_0, \ldots, t_{q-1}) \). Then the complete weight enumerator of \( \mathcal{A} \) is
\[
  W_\mathcal{A}(z_0, \ldots, z_{q-1}) = \sum_t A(t) z_0^{t_0} \cdots z_{q-1}^{t_{q-1}}
  = \sum_{c \in \mathcal{A}} z_0^{s_0} \cdots z_{q-1}^{s_{q-1}}.
\]

where \( z_0, z_1, \ldots, z_{q-1} \) is a set of \( q \) commuting indeterminants and the indeterminant \( z_i \) corresponds to the element \( \omega_i \).
5.2 On the Probability of Post Decoder Events for (216,175) Binary Code

For the decoding algorithm described in chapter 3 the various parameters of interest are

1. The probability of correct decoding, \( P_{CD} \)
2. The probability of incorrect decoding, \( P_{ICD} \)
3. The probability of decoding failure, \( P_{F} \)
4. The output symbol error rate, \( P_{SE} \) and
5. The output bit error rate, \( BER \).

It was shown in chapter 2 that all the 8-tuples in the binary code obtained as a result of the mapping, are of even weight and a received 8-tuple having odd weight is termed as an erasure. Let the 8-tuples having odd numbers of 1's be represented by indeterminants \( z_1^*, z_2^*, \ldots, z_{128}^* \).

It can be readily observed that on a BSC, the probability that the all 0 8-tuple will get converted to any one of these is given by

\[
P_E = \sum_{i=0}^{3} \binom{8}{2i+1} P^{2i+1} Q^{8-(2i+1)}
\]  

(5.2.1)

\( P \) being the cross over probability.

In the following analysis, the patterns corresponding to various events are derived and the probability of occurrence of each pattern can be calculated by replacing each of the \( z_i^* \) by \( P_{i}^{\ell_i} Q^{8-\ell_i} \), where \( \ell_i \) is the number of 1's in the binary 8-tuple representation of \( z_i^* \). Also note that \( q = 128 \).
5.2.1 The Probability of Correct Decoding, $P_{CD}$

The received vector is decoded correctly if

(a) It is same as the all 0 code word.

(b) It has only one non-zero symbol and the symbol has even weight (one error)

(c) It has only one non-zero symbol and that symbol has odd weight (one erasure)

(d) It has two non-zero symbols and both have odd weight (two erasures).

Pattern corresponding to event (a) is given by $z_0^n$. Similarly, patterns corresponding to events (b), (c) and (d) are given by

$$n(z_1 + z_2 + \cdots + z_{127})z_0^{n-1}$$

$$n(z_1^* + z_2^* + \cdots + z_{128}^*)z_0^{n-1}$$

and

$$\binom{n}{2}(z_1^* + z_2^* + \cdots + z_{128}^*)^2 z_0^{n-2}.$$

Hence, all the received patterns that are decoded correctly are

$$z_0^n + n(z_1 + \cdots + z_{127})z_0^{n-1} + n(z_1^* + \cdots + z_{128}^*)z_0^{n-1} + \binom{n}{2}(z_1^* + \cdots + z_{128}^*)^2 z_0^{n-2},$$

where $n$ is the block length of the code.  \hfill (5.2.2)

5.2.2 The Probability of Incorrect Decoding, $P_{ICD}$

An incorrect decoding takes place if the received vector leads to one of the following events

(a) It is same as a non-zero code word

(b) It differs from a non-zero code word in one position and the symbol received in that position has even weight.
(c) It differs from a non-zero code word in one position and the symbol received in that position has odd weight.

(d) It differs from a non-zero code word in two positions and the symbols received in these positions have odd weight.

Let a code word be represented as

\[ c = z_0 ^{s_0} z_1 ^{s_1} \cdots z_{127} ^{s_{127}} \]  \hspace{1cm} (5.2.3)

and let the patterns corresponding to the received vectors that will be decoded to this code word be represented by \( e \). For each of the above given error events, the patterns are obtained as follows.

Event (a). It is same as the code word and, therefore, is given by

\[ e = z_0 ^{s_0} z_1 ^{s_1} \cdots z_{127} ^{s_{127}} \]

It is clear that all the \( 0 \) received word is decoded correctly and hence all the patterns that lead to this error event are

\[ e = \sum_{c \in \mathcal{C}} c - z_0 ^n \]  \hspace{1cm} (5.2.4)

Event (b). All the patterns that differ from the code word in a position that corresponds to \( z_0 \) in a code word say, can be represented as

\[ \begin{array}{cccc}
  s_0 ^{-1} & s_1 + 1 & \cdots & s_{127} \\
  z_0 & z_1 & \cdots & z_{127} \\
\end{array} \]  

\[ + \begin{array}{cccc}
  s_0 ^{-1} & s_1 & s_2 + 1 \cdots & s_{127} \\
  z_0 & z_1 & z_2 & \cdots & z_{127} \\
\end{array} \]

\[ + \cdots + \begin{array}{cccc}
  s_0 ^{-1} & s_1 \cdots & s_{127} + 1 \\
  z_0 & z_1 \cdots & z_{127} \\
\end{array} \]

---

1 This event will also lead to an undetectable error, if this coding scheme is used for the purpose of error detection only.
\[ e = (z_1 + z_2 + \cdots + z_{127})c/z_0 \]

and there are \( s_0 \) of such patterns.

Hence, it can be shown that all the patterns that differ from a code word \( c \) in one place are

\[
e = s_0(z_1 + z_2 + \cdots + z_{127})c/z_0 \\
+ s_1(z_0 + z_2 + \cdots + z_{127})c/z_1 \\
+ \cdots \\
+ s_{127}(z_0 + z_1 + \cdots + z_{126})c/z_{127} \\
= \sum_{i=0}^{127} s_i(z_0 + \cdots + z_{i-1} + z_{i+1} + \cdots + z_{127})c/z_i \\
= \sum_{i=0}^{127} s_i(z_0 + z_1 + \cdots + z_{127})c/z_i - \sum_{i=0}^{127} s_i c \tag{5.2.5}
\]

Clearly

\[
\sum_{i=0}^{127} s_i = n,
\]

and, therefore, Equation (5.2.5) can be written as

\[
e = (z_0 + z_1 + \cdots + z_{127}) \sum_{i=0}^{127} s_i c/z_i - n c.
\]

Since the received word that has only one non-zero symbol is decoded correctly, all the patterns that lead to this error event are

\[
e = \sum_{c \in \mathbb{C}^n} (z_0 + z_1 + \cdots + z_{127}) \sum_{i=0}^{127} s_i c/z_i - n c \tag{5.2.6}
\]

\[ - n(z_1 + z_2 + \cdots + z_{127})z_0^{n-1} \]
Event (c). The received vector can differ from a code word at any one of the positions corresponding to \( z_0, z_1, \ldots, z_{127} \) in the code word and the symbol received at that position can be any one of \( z_1^*, z_2^*, \ldots, z_{128}^* \). A similar analysis can be performed as above and it can be shown that all such patterns that lead to this error event are

\[
e = \sum_{c \in \mathbb{A}} (z_1^* + z_2^* + \cdots + z_{128}^*) \sum_{i=0}^{127} s_i c / z_i \sum_{i=0}^{n-1} -n(z_1^* + z_2^* + \cdots + z_{128}^*) z_0^{-1}
\]

(5.2.7)

Event (d). For a code word represented by Equation (5.2.3), the patterns that differ from the code word in two positions and have symbols of odd weight in those positions can be represented as

\[
(z_1^* + z_2^* + \cdots + z_{128}^*)^2 \sum_{i=0}^{127} s_i (s_i - 1) z_0 s_1 \cdots z_i \cdots z_{127}
\]

\[
+ (z_1^* + z_2^* + \cdots + z_{128}^*)^2 \sum_{i=0}^{127} \sum_{j=0}^{126} s_j s_i z_0 s_1 \cdots z_i \cdots z_j \cdots z_{127}
\]

\[
+ (z_1^* + z_2^* + \cdots + z_{128}^*)^2 \sum_{i=0}^{127} \sum_{j=0}^{126} s_j s_i z_0 s_1 \cdots z_i \cdots z_j \cdots z_{127}
\]

where \( s_i^* = s_i - \delta(i-j), \delta(x) \) being the delta function.

From Equation (5.2.3), it can be seen that

\[
s_0 s_1 \cdots s_{j-1} s_j \cdots s_{127} z_0 z_1 \cdots z_i \cdots z_j \cdots z_{127} = c / (z_j z_i).
\]

Therefore, all the patterns that lead to this error event are

\[
e = (z_1^* + z_2^* + \cdots + z_{128}^*)^2 \sum_{c \in \mathbb{A}} \sum_{j=0}^{127} \sum_{i=0}^{127} s_j s_i c / (z_j z_i)
\]

\[
- \frac{n(n-1)}{2} (z_1^* + z_2^* + \cdots + z_{128}^*)^2 z_0^{-2}
\]

(5.2.8)
Hence, all the received words that would lead to incorrect decoding can be represented by the sum of expressions (5.2.4), (5.2.6), (5.2.7) and (5.2.8).

5.2.3 The Probability of Decoding Failure, \( P_F \)

This is an event when the decoder detects an error but it is beyond its correcting capability. The probability of correct decoding, incorrect decoding and decoding failure are related as

\[
P_F + P_{ICD} + P_{CD} = 1
\]

or

\[
P_F = 1 - P_{ICD} - P_{CD}
\] (5.2.9)

5.2.4 Post Decoder Symbol Error Rate, \( P_{SE} \)

The number of non-zero symbols for a code word represented by Equation (5.2.3) is \( s_1 + s_2 + \cdots + s_{127} \). Also

\[
s_0 + s_1 + \cdots + s_{127} = n
\]

or

\[
s_1 + s_2 + \cdots + s_{127} = n - s_0
\]

If \( e \) represents all the error patterns that are incorrectly decoded to a code word \( c \), then the post decoder symbol error rate is given by

\[
P_{SE} = \frac{1}{n} \sum_{c \in \mathcal{A}} (s_1 + s_2 + \cdots + s_{127})e
\]

\[
= \frac{1}{n} \sum_{c \in \mathcal{A}} (n - s_0)e
\]

\[
= \sum_{c \in \mathcal{A}} e - \frac{1}{n} \sum_{c \in \mathcal{A}} s_0e
\]
There are four error events as described above, that lead to incorrect decoding and, therefore, would also contribute to the symbol error rate. Thus, the post decoder symbol error rate is the sum of the symbol error rates due to each of the four error events. Let these be represented by $P_{SE}(a), P_{SE}(b), P_{SE}(c), P_{SE}(d)$ respectively.

Event (a)

The pattern $e$ for any code word $c$ is the same as the code word itself. $P_{SE}(a)$ is, then, given by

$$P_{SE}(a) = \sum_{c \in \mathcal{H}} c - \frac{1}{n} \sum_{c \in \mathcal{H}} s_0 c$$  \hspace{1cm} (5.2.10)

Event (b)

The error patterns that lead to this event are given by Equation (5.2.6) and it can be shown that $P_{SE}(b)$ is

$$P_{SE}(b) = \sum_{c \in \mathcal{H}} [(z_0 + z_1 + \cdots + z_{127}) \sum_{i=0}^{127} s_i c/z_i - nc$$

$$- \frac{1}{n} [(z_0 + z_1 + \cdots + z_{127}) \sum_{i=0}^{127} s_0 s_i c/z_i - ns_0 c]$$  \hspace{1cm} (5.2.11)

Event (c)

The error patterns that lead to this event are given by Equation (5.2.7) and $P_{SE}(c)$ can be expressed as

$$P_{SE}(c) = \sum_{c \in \mathcal{H}} [(z_1^* + z_2^* + \cdots + z_{127}^*) \sum_{i=0}^{127} s_i c/z_i$$

$$- \frac{1}{n} (z_1^* + z_2^* + \cdots + z_{127}^*) \sum_{i=0}^{127} s_0 s_i c/z_i]$$  \hspace{1cm} (5.2.12)
Event (d)

Again, the error patterns that lead to this event are given by Equation (5.2.8) and \( P_{SE}(d) \) can be written as

\[
P_{SE}(d) = \sum_{c \in \mathcal{C}} \left[ (z_1^* + z_2^* + \cdots + z_{128}^*)^2 \sum_{j=0}^{127} \sum_{i=0}^{127} \frac{1}{2^n} s_j s_i^* c/(z_j z_i) \right]
\]

(5.2.13)

\[
- (z_1^* + z_2^* + \cdots + z_{128}^*)^2 \sum_{j=0}^{127} \sum_{i=0}^{127} \frac{1}{2^n} s_j s_i^* c/(z_j z_i)
\]

and finally

\[
P_{SE} = P_{SE}(a) + P_{SE}(b) + P_{SE}(c) + P_{SE}(d)
\]

5.2.4 Post Decoder Bit Error Rate, BER

For every received vector decoded to a code word represented by Equation (5.2.3), the number of bits in error is given by

\[
s_{1,1} + s_{2,2} + \cdots + s_{127,127}
\]

where \( \omega_i \) is the number of 1's in the binary 8-tuple representation of the element \( \omega_i \).

Thus, if \( e \) represents all the patterns that are incorrectly decoded to this code word, then BER is

\[
BER = \frac{1}{8^n} \sum_{c \in \mathcal{C}} (s_{1,1} + s_{2,2} + \cdots + s_{127,127})^e
\]

It is an extremely difficult task to evaluate such an expression. However, a very close bound may be obtained as

\[
BER = P_{SE}.
\]

This bound is obtained by noting that the maximum value that any \( \omega_i \) takes is 8.
5.2.5 Note

The code to be analysed is a (27,25) RS code defined over GF(2^7), where the binary 8-tuples forming a vector space of dimension 7 are used to represent elements of GF(2^7). There are no closed form expressions for complete weight enumerator of such a code. Also, as the total number of code words in this code is \(4.789 \times 10^{52}\), it is not possible to generate the complete weight enumerator of this code on the computer and then evaluate the probability of all the error patterns for each of the code words.

It is further noted that if the probability of 0 symbol being received as a non-zero symbol was the same for all the non-zero symbols, then the Hamming weight enumerator of the code was sufficient for the statistical performance analysis of the code. The Hamming weight enumerator of this code is given by the Equation (4.2.2).

Hence we have to look for alternative ways of computing the expressions derived here. One such method is to use MacWilliams theorem for complete weight enumerator. The dual code of (27,25) code is (27,2) code and has only 16384 code words. Therefore, it is possible to generate the complete weight distribution of the dual code on the computer.

In the following section, MacWilliams theorem for complete weight enumerator and the related theory is covered in brief.
5.3 MacWilliams Theorem for Complete Weight Enumerator

5.3.1 Characters of GF(q)

Any element \( \beta \) of GF(q), \( q = 2^m \) can be written in the form

\[
\beta = \beta_0 + \beta_1 \alpha + \cdots + \beta_{m-1} \alpha^{m-1}
\]

or equivalently as an \( m \)-tuple

\[
\beta = (\beta_0, \beta_1, \ldots, \beta_{m-1}),
\]

where \( \alpha \) is a primitive element of GF(q) and \( 0 \leq \beta_i \leq 1 \). Let \( \xi \) be a complex number \( e^{2\pi i/2} \). This is a primitive \( 2^{nd} \) root of unity, i.e. \( \xi^2 = e^{2\pi i} = 1 \), while \( \xi^k \neq 1 \) for \( 0 < k < 2 \). It implies \( \xi = -1 \).

Definition. For each \( \beta = (\beta_0, \beta_1, \ldots, \beta_{m-1}) \) of GF(q), define \( x_\beta \) to be the complex valued mapping defined on GF(q) by

\[
x_\beta(v) = \xi^{\beta_0 v_0 + \cdots + \beta_{m-1} v_{m-1}}
\]

for \( v = (v_0, \ldots, v_{m-1}) \in GF(q) \). \( x_\beta \) is called a character of GF(q).

It can be easily shown that

(i) \( x_\beta(v) = x_\gamma(\beta) \) for all \( \beta, v \in GF(q) \)

(ii) \( x_\beta(v + v') = x_\beta(v) \cdot x_\beta(v') \) for all \( \beta, v, v' \in GF(q) \)

Thus, \( x_\beta \) is a homomorphism from the additive group of GF(q) into the multiplicative group of complex numbers of magnitude 1.

(iii) \( x_{\beta + \beta'}(v) = x_\beta(v) \cdot x_{\beta'}(v) \) for all \( \beta, \beta', v \in GF(q) \)

Thus the set of all \( q \) characters \( x_\beta \) form a group which is isomorphic to the additive group of GF(q).
To state MacWilliams theorem [1], any one of the characters $x_\beta$ with $\beta \neq 0$ is selected, say $\beta = 1$, i.e. the character $x_1$ defined by

$$x_1(v) = \xi^{v_0} \text{ for } v = (v_0, \ldots, v_{m-1}) \in GF(q)$$

5.3.2 MacWilliams Theorem for Complete Weight Enumerator [13]

If $\mathcal{A}$ is a linear $(n,k)$ code over $GF(q)$ with complete weight enumerator $W_{\mathcal{A}}$, the complete weight enumerator of the dual code $W_{\mathcal{A}^\perp}$ is

$$W_{\mathcal{A}^\perp}(z_0, \ldots, z_{q-1}) = \frac{1}{|\mathcal{A}|} W_{\mathcal{A}} \left( \sum_{i=0}^{q-1} x_1(\omega_i)z_i, \ldots, \sum_{i=0}^{q-1} x_1(\omega_i)z_i, \ldots \right)$$

(5.3.1)

where $|\mathcal{A}| = q^k$,

or alternatively

$$W_{\mathcal{A}^\perp}(z_0, \ldots, z_{q-1}) = \frac{1}{|\mathcal{A}|^2} W_{\mathcal{A}^\perp} \left( \sum_{i=0}^{q-1} x_1(\omega_i)z_i, \ldots, \sum_{i=0}^{q-1} x_1(\omega_i)z_i, \ldots \right)$$

(5.3.2)

where $|\mathcal{A}| = q^{n-k}$.

Let

$$y_0 = \sum_{i=0}^{q-1} x_1(\omega_i)z_i$$

$$\ldots$$

$$y_{q-1} = \sum_{i=0}^{q-1} x_1(\omega_{q-1})z_i$$

(5.3.3)

and we can write Equation (5.3.2) as

$$W_{\mathcal{A}^\perp}(z_0, \ldots, z_{q-1}) = \frac{1}{|\mathcal{A}|} W_{\mathcal{A}^\perp} (y_0, \ldots, y_{q-1})$$

(5.3.4)

In our case
\[ q = 128, \]
\[ n-k = 2. \]

Hence, the weight enumerator of the original code is related to the weight enumerator of the dual code as

\[ W_q(z_0, \ldots, z_{127}) = \frac{1}{128^2} W_q(y_0, \ldots, y_{127}). \]

Thus, if \( c^1 \) is an arbitrary code word in the dual code with \( \text{comp}(c^1) = (t_0, \ldots, t_{127}) \), then the weight distribution of the original code is

\[ \sum_{c \in \mathbb{F}_2^{128}} s_0 c_0 s_1 c_1 \ldots s_{127} c_{127} = \frac{1}{128^2} \sum_{c \in \mathbb{F}_2^{128}} t_0 y_0 t_1 y_1 \ldots t_{127} y_{127}. \quad (5.3.5) \]

5.4 Further Analysis

Though the complete weight enumerator of the dual code can be generated on the computer, it is to be observed that it is not possible to find the composition of each and every code word in the original code by using Equation (5.3.5). Also note that expressions derived for various events in section (5.2) are in terms of the composition variables \( s_0, s_1, \ldots, s_{127} \). Hence, we should find expressions for these events that are functions of the variables \( z_0 z_1, \ldots, z_{127} \) only and where the composition variables \( s_0, s_1, \ldots, s_{127} \) which define the composition of each of the code words, do not appear explicitly. Now, using Equation (5.2.3), we can show that

\[ s_j c / z_1 = \frac{a c}{a z_1} \]

and

\[ s_j s_i^* c / (z_j z_1) = \frac{a^2 c}{a z_j a z_1}. \quad (5.4.1) \]
Using above equation, Equations (5.2.6), (5.2.7), (5.2.8), (5.2.10),
(5.2.11), (5.2.12) and (5.2.13) can be rewritten as

\[ e = \sum_{c \in \&} (z_0 + \cdots + z_{127}) \sum_{i=0}^{127} \frac{ac}{az_i} - nc \]
\[ - n(z_1 + \cdots + z_{127})z_0^{n-1} \]
\[ = \sum_{c \in \&} (z_0 + \cdots + z_{128}) \sum_{i=0}^{127} \frac{ac}{az_i} \]
\[ - n(z_1 + \cdots + z_{128})z_0^{n-1} \]
\[ e = \frac{(z_1^* + \cdots + z_{128}^*)^2}{2} \sum_{c \in \&} \sum_{j=0}^{127} \sum_{i=0}^{127} \frac{a^2 c}{az_jaz_i} \]
\[ - \frac{n(n-1)}{2} \frac{(z_1^* + \cdots + z_{128}^*)^2}{2} z_0^{n-2} \]
\[ p_{SE}(a) = \sum_{c \in \&} c - \frac{1}{n} \sum_{c \in \&} z_0 \frac{ac}{az_0} \]
\[ p_{SE}(b) = \sum_{c \in \&} \left[ (z_0 + \cdots + z_{127}) \sum_{i=0}^{127} \frac{ac}{az_i} - nc \right. \]
\[ - \frac{1}{n} \left\{ (z_0 + \cdots + z_{127}) \left( \frac{ac}{az_0} + z_0 \frac{a}{az_0} \left( \sum_{i=0}^{127} \frac{ac}{az_i} \right) \right) - nz_0 \frac{ac}{az_0} \right\} \]
\[ p_{SE}(c) = (z_1^* + \cdots + z_{128}) \sum_{c \in \&} \left[ \sum_{i=0}^{127} \frac{ac}{az_i} - \frac{1}{n} \left( \frac{ac}{az_0} + z_0 \frac{a}{az_0} \left( \sum_{i=0}^{127} \frac{ac}{az_i} \right) \right) \right] \]
\[ \left. + \frac{a^2 c}{az_0} \right\} \]
\[ p_{SE}(d) = \frac{1}{2} (z_1^* + \cdots + z_{128})^2 \sum_{c \in \&} \left[ \sum_{j=0}^{127} \sum_{i=0}^{127} \frac{a^2 c}{az_jaz_i} \right. \]
\[ - \frac{1}{n} \left\{ z_0 \frac{a}{az_0} \left( \sum_{j=0}^{127} \sum_{i=0}^{127} \frac{a^2 c}{az_jaz_i} \right) + 2 \frac{a}{az_0} \left( \sum_{i=0}^{127} \frac{ac}{az_i} \right) \right\} \]
A close examination of the above equations reveals that, we have to evaluate expressions of the type

\[(i) \sum_{c \in \&} 127 \sum_{i=0}^{\frac{3c}{aZ_i}} \frac{ac}{aZ_i} \]

\[(ii) \sum_{c \in \&} 127 \sum_{j=0}^{\frac{2c}{aZ_i}} \frac{a^2}{aZ_i} \]

The corresponding expressions are obtained in terms of the weight distribution of the dual code by using Equation (5.3.5). This is done as follows.

\[(i) \sum_{c \in \&} 127 \sum_{i=0}^{\frac{3c}{aZ_i}} \frac{ac}{aZ_i} = \sum_{c \in \&} \frac{3}{aZ_i} (\sum_{c \in \&} c)\]

Using Equation (5.3.5), we get

\[
\frac{127}{128^2} \sum_{i=0}^{127} \frac{3}{aZ_i} \left( \sum_{c \in \&} c \right) = \frac{127}{128^2} \sum_{i=0}^{127} \frac{3}{aZ_i} \left( \frac{1}{128^2} \sum_{c \in \&} t_0 \cdots t_{127} \right)
\]

\[
= \frac{1}{128^2} \sum_{c \in \&} \sum_{i=0}^{127} \frac{3}{aZ_i} (y_0 \cdots y_{127})
\]

\[
= \frac{1}{128^2} \sum_{c \in \&} \sum_{i=0}^{127} \sum_{j=0}^{t_j} t_j y_0 \cdots y_j \cdots y_{127} \frac{\partial y_j}{\partial Z_i}
\]

Using Equation (5.3.3), we get

\[
\frac{\partial y_j}{\partial Z_i} = x_j(\omega_j \omega_i).
\]

Substituting \(\frac{\partial y_j}{\partial Z_i}\) from the above equation, we can write
\[
\sum_{c \in \&} \sum_{i=0}^{127} \frac{ac}{\bar{z}_i} = \frac{1}{128^2} \sum_{c \in \&^4} \sum_{i=0}^{127} \sum_{j=0}^{127} t_j y_0 \ldots y_{j-1} \ldots y_{127} x_1(\omega_j \omega_i)
\]
\[
= \frac{1}{128^2} \sum_{c \in \&^4} \sum_{j=0}^{127} t_j y_0 \ldots y_{j-1} \ldots y_{127} \sum_{i=0}^{127} x_1(\omega_j \omega_i)
\]
\[
(5.4.9)
\]

From the theory of group characters [13]

\[
\sum_{i=0}^{127} x_1(\omega_j \omega_i) = 128 \text{ if } j = 0
\]

\[
= 0 \text{ otherwise}
\]

Using the above result, expression (5.4.9) can be reduced to

\[
\sum_{c \in \&} \sum_{i=0}^{127} \frac{ac}{\bar{z}_i} = \frac{1}{128^2} \sum_{c \in \&^4} \sum_{i=0}^{128} t_0 c^i / y_0,
\]
\[
(5.4.10)
\]

where \( c^i = y_0 y_1 \ldots y_{127} \).

(ii) \[
\sum_{c \in \&} \sum_{j=0}^{127} \frac{a^2 c}{\bar{z}_j \bar{z}_i} = \sum_{j=0}^{127} \sum_{i=0}^{a^2} \frac{a^2}{\bar{z}_j \bar{z}_i} \sum_{c \in \&^4} \sum_{i=0}^{127} t_0 \ldots t_{127}
\]
\[
= \sum_{j=0}^{127} \sum_{i=0}^{127} \frac{a^2}{\bar{z}_j \bar{z}_i} \left( \frac{1}{128^2} \sum_{c \in \&^4} \sum_{i=0}^{127} y_0 \ldots y_{127} \right)
\]
\[
= \frac{1}{128^2} \sum_{c \in \&^4} \sum_{j=0}^{127} \frac{a}{\bar{z}_j} \left( \sum_{i=0}^{127} \frac{a}{\bar{z}_i} (y_0 \ldots y_{127}) \right)
\]
\[
(5.4.11)
\]

It was shown in part (i) that

\[
\sum_{i=0}^{127} \frac{a}{\bar{z}_i} (y_0 \ldots y_{127}) = 128 t_0 t_0^{-1} t_1 \ldots t_{127},
\]

and, therefore, Equation (5.4.11) can be simplified to
\[
\frac{1}{128^2} \sum_{c \in \mathbb{A}} \sum_{j=0}^{127} \frac{2}{3} \text{e}_{j} (128 \cdot t_0 \cdot y_0 \cdot t_0^{-1} \cdot y_1 \cdot \cdots \cdot t_{127}) \\
= \frac{1}{128^2} \sum_{c \in \mathbb{A}} 128 \cdot 128 \cdot t_0(t_0 - 1) y_0 t_0^{-2} y_1 \cdots y_{127}
\]

(5.4.12)

Since the dual of a maximum distance separable code is also a maximum distance separable code, the dual code of (27,25) RS code which is (27,2) code has the following Hamming weight distribution

1 code word has all \( n \) symbols as zeros (all 0 code word), 
\( t_0 = n \)

3429 code words have only one 0 symbol, \( t_0 = 1 \)

12964 code words have no 0 symbols, \( t_0 = 0 \)

Note that \( t_0 \) is the number of zero symbols in code word of the dual code and, therefore, the term \( t_0(t_0 - 1) \) is non-zero only for the all zero code word, i.e. for \( t_0 = n \). Consequently,

\[
\sum_{c \in \mathbb{A}} \sum_{j=0}^{127} \frac{2}{3} c_{j} = n(n - 1) y_0
\]

(5.4.13)

Equations (5.4.10) and (5.4.13) are used to substitute for the corresponding summation in the expressions for \( P_{ICD} \) and \( P_{SE} \). Finally, all the patterns that correspond to each of the events of interest for the (27,25) code are given as below

(1) \( P_{CD} = z_0^{27} \binom{27}{1} (z_1 + z_2 + \cdots + z_{127})z_0^{26} + \binom{27}{1} (z_1^* + \cdots + z_{128}^*)z_0^{26} \)

\[+ \binom{27}{2} (z_1^* + \cdots + z_{128}^*)^2 z_0^{25} \]
\[ P_{\text{ICD}} = \frac{1}{128^2} \sum_{c^i \in \mathcal{A}} \left[ 128 t_0 c^i - 26 c^i + 128 (z_1^* + \cdots + z_{128}^*) t_0 c^i / y_0 \right] \]
\[ + \binom{27}{2} (z_1^* + \cdots + z_{128}^*)^2 25 y_0 - P_{\text{CD}} \]

(3) \[ P_f = 1 - P_{\text{CD}} - P_{\text{ICD}} \]

(4) \[ P_{\text{SE}} = P_{\text{ICD}} + P_{\text{CD}} + \frac{1}{27.128^2} \sum_{c^i \in \mathcal{A}} \left[ (25z_0 - z_1 - \cdots - z_{127}) \left( \sum_{i=0}^{127} \frac{t_i}{y_i} \right) c^i \right. \]
\[ - 128.z_0 t_0 \left( \frac{t_0-1}{y_0} + \frac{t_1}{y_1} + \cdots + \frac{t_{127}}{y_{127}} \right) c^i - (z_1^* + \cdots + z_{128}^*) \left( \sum_{i=0}^{127} \frac{t_i}{y_i} \right) c^i \]
\[ - 128.z_0 (z_1^* + \cdots + z_{128}^*) t_0 \left( \frac{t_0-1}{y_0} + \frac{t_1}{y_1} + \cdots + \frac{t_{127}}{y_{127}} \right) c^i / y_0 \]
\[ - 128(z_1^* + \cdots + z_{128}^*)^2 t_0 \left( \frac{t_0-1}{y_0} + \frac{t_1}{y_1} + \cdots + \frac{t_{127}}{y_{127}} \right) c^i / y_0 \]
\[ - \frac{25}{2} (z_1^* + \cdots + z_{128}^*)^2 z_0 y_0^2 \]

These expressions are computed for different values of the input bit error rate \( P \). Plots of \( P_{\text{ud}}, P_{\text{CD}}, P_{\text{ICD}}, P_f \) and \( P_{\text{SE}} \) versus \( P \) are given in Figures (5.2), (5.3), (5.4), (5.5) and (5.6) respectively.

5.5 A Note on the Performance of (224,175) Binary Code

At this moment complete performance evaluation of (224,175) binary code obtained from (28,25) RS code does not seem to be computationally feasible on a binary symmetric channel. However, the probability of correct decoding \( P_{\text{CD}} \) for this code can be calculated as follows.

This code can decode correctly all patterns of received vector patterns given in section (5.2.1). It can also decode correctly those received vectors that
have three non-zero symbols and all three symbols have odd weight (three erasures)

- have two non-zero symbols with one of the two having odd weight and the other one having even weight (one erasure, one error).

These vectors can be represented by the patterns

\[
\binom{n}{3} (z_1^* + \cdots + z_{128}^*)^3 z_0^{n-3}
\]

\[
+ n(n-1)(z_1 + \cdots + z_{127})(z_1^* + \cdots + z_{128}^*) z_0^{n-2}
\]

Hence for \((224,175)\) binary code, all the received patterns that are decoded correctly are

\[
z_0^{28} + 28(z_1 + \cdots + z_{127})z_0^{27} + 28(z_1^* + \cdots + z_{128}^*)z_0^{27}
\]

\[
+ \binom{28}{2} (z_1^* + \cdots + z_{128}^*)^2 z_0^{26} + \binom{28}{3} (z_1^* + \cdots + z_{128}^*)^3 z_0^{25}
\]

\[
+ 28.27 (z_1 + \cdots + z_{127})(z_1^* + \cdots + z_{128}^*) z_0^{26}
\]

A plot of \(P_{CD}\) versus the input bit error rate \(P\) is given in Figure (5.3) for this code.

5.6 Discussion of Results

The binary code obtained as a result of mapping \((27,25)\) RS code defined over \(GF(2^7)\) into a binary code, is a \((216,175)\) code and has a minimum distance \(d_b\) equal to 6. Therefore, it should correct all the possible single and double errors in a received word. This, indeed, is the case for the decoding algorithm described in chapter 3. The code can also decode correctly certain patterns of errors in more than two bits. Such a scheme can find a very wide application in digital
communication system where the errors occur randomly or cluster in bursts.

Although the exact performance analysis of the (224,175) binary code obtained as a result of mapping (28,25) RS code defined as $\text{GF}(2^7)$, is not feasible, yet we are assured of a better performance as compared to the (216,175) binary code since the (224,175) binary code has a minimum distance $d_\min$ of 6 and the decoding algorithm described in chapter 3 corrects any three or fewer bits in error as well as some error patterns of more than three bits.

Both of the high rate codes analysed here compare well with the codes described in [14] and chapters 2 and 3 of [7] for byte oriented information systems. It must be pointed that the codes detailed in [14] and [7] are strictly random error correcting while the codes given here can correct both random as well as bursts of error. However, the latter codes require one more byte of redundancy as compared to the former code. For example, the (216,175) code based on the scheme given in [7] can correct all single errors, double errors and any triple error pattern occurring in three distinct information bytes, whereas, the (224,175) binary code described in this thesis can decode correctly all the possible error patterns of single, double and triple errors and certain restricted error patterns of more than three bits.

If further improvement in performance is required for a system, then a code of higher minimum distance can be derived from the coding scheme described in chapter 2. The formulation of the code given in

---

1 It is 70 percent of all the possible triple error patterns.
[7] needs to be explored further for the case of more than two bytes of redundancy and the properties of the scheme are not known completely.

Hence if a high rate coding scheme is required for a byte oriented information system and the channel introduces only random errors during the transmission, then the scheme given in [7] may be desirable for the purpose of error control but if the channel characteristics are not determined completely or if it introduces both random and burst errors, then the coding scheme analysed in this thesis, may be preferred.
FIG. 5.2: PROBABILITY OF UNDETECTED ERROR VS. INPUT BIT ERROR RATE FOR (216,175) BINARY CODE
FIG. 5.3. PROBABILITY OF CORRECT DECODING Vs.
INPUT BIT ERROR RATE FOR BINARY CODES.
FIG. 5.4. PROBABILITY OF INCORRECT DECODING Vs. INPUT BIT ERROR RATE FOR (216, 175) BINARY CODE
FIG. 5.5. PROBABILITY OF DECODING FAILURE VS. INPUT BIT ERROR RATE FOR (216,175) BINARY CODE
FIG. 5.6: POST DECODER SYMBOL ERROR RATE vs. INPUT BIT ERROR RATE FOR (216,175) BINARY CODE
CHAPTER 6

Conclusions

The main objective of this thesis has been to design a coding scheme for even parity byte information systems where the number of parity bits added to any block of information bytes are a multiple of 8 and the parity bit present in each of the information bytes is not allowed to be altered. Although codes of any rate can be obtained by employing the coding scheme presented here, the emphasis has been on the development and analysis of high rate codes.

In chapter 2, the mapping of codes defined over $GF(2^m)$ into binary codes has been studied and it is shown that the set of all possible even parity 8-tuples forms a vector space of dimension 7. A RS code defined over $GF(2^7)$ was considered and the procedure to get the generator matrix in the systematic form for the binary codes obtained by mapping the RS codes was outlined. The parity bytes added to the information bytes, are shown to have even parity. A shift register implementation for the coding scheme was illustrated.

Since the emphasis is on high rate codes, a decoding algorithm was presented only for the cases when the RS code, used to derive the binary code, had a minimum distance of 3 and 4. The decoding algorithm makes use of the parity bit present in each of the received bytes. As it is in closed form, the decoding can be implemented by digital hardware that is capable of operating at very high speed of data transmission.

A channel can be modelled as either a q-ary symmetric channel ($q \neq 2$) or a binary symmetric channel, depending on the modulation.
scheme that is used to transmit the data. Both models were considered for the statistical performance analysis of the binary code. However, it was observed that the analysis of the code on the binary symmetric channel is very complicated and, therefore, it was performed only for the (216,175) binary code. The analysis of the (224,175) binary code was restricted to the evaluation of the probability of correct decoding.

The coding scheme derived in this thesis has the potential of combating random and burst errors that normally occur in digital communication systems. It is clear from the plots for the probability of various post decoder error events given in chapters 4 and 5 that using these codes can lead to significant improvement in the overall system performance.
BIBLIOGRAPHY


7. P.E. Allard, V.K. Bhargava and G.E. Séguin, "Realization, Economic and Performance Analysis of Error-Correcting Codes and ARQ Systems for Broadcast Telidon and Other Videotex Systems", Research Report, Department of Electrical Engineering, Concordia University, Montreal, June 1981. (Prepared for the Canadian Dept. of Communications under DSS contract No. OSU80-00133.)


APPENDIX A

Table of Elements of GF(2^7)

In this appendix, a table of \( i \) with the binary 8-tuple representation of \( \alpha^i \) for the basis chosen in Equation (2.2.1) is given, where \( \alpha \) is a primitive element of \( GF(2^7) \). We have used the recursion \( \alpha^{2^7} = 1 + \alpha^3 \).

<table>
<thead>
<tr>
<th>( i )</th>
<th>10000001</th>
<th>01101100</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00000101</td>
<td>11010001</td>
</tr>
<tr>
<td>2</td>
<td>00001000</td>
<td>00101011</td>
</tr>
<tr>
<td>3</td>
<td>00010100</td>
<td>01001000</td>
</tr>
<tr>
<td>4</td>
<td>00100000</td>
<td>11010001</td>
</tr>
<tr>
<td>5</td>
<td>11000000</td>
<td>00001101</td>
</tr>
<tr>
<td>6</td>
<td>00001001</td>
<td>00111010</td>
</tr>
<tr>
<td>7</td>
<td>00010010</td>
<td>01110000</td>
</tr>
<tr>
<td>8</td>
<td>00100100</td>
<td>11100001</td>
</tr>
<tr>
<td>9</td>
<td>01000100</td>
<td>01001011</td>
</tr>
<tr>
<td>10</td>
<td>01011001</td>
<td>10011111</td>
</tr>
<tr>
<td>11</td>
<td>10111001</td>
<td>11111100</td>
</tr>
<tr>
<td>12</td>
<td>11100000</td>
<td>01110001</td>
</tr>
<tr>
<td>13</td>
<td>11100101</td>
<td>11101001</td>
</tr>
<tr>
<td>14</td>
<td>01001010</td>
<td>01101111</td>
</tr>
<tr>
<td>15</td>
<td>10010011</td>
<td>10101000</td>
</tr>
<tr>
<td>16</td>
<td>10010110</td>
<td>01001111</td>
</tr>
<tr>
<td>17</td>
<td>10110011</td>
<td>01011101</td>
</tr>
<tr>
<td>18</td>
<td>11010111</td>
<td>01011000</td>
</tr>
<tr>
<td>19</td>
<td>00111001</td>
<td>00100100</td>
</tr>
<tr>
<td>20</td>
<td>01100010</td>
<td>01100001</td>
</tr>
<tr>
<td>21</td>
<td>11011101</td>
<td>01100110</td>
</tr>
<tr>
<td>22</td>
<td>11100111</td>
<td>01110010</td>
</tr>
<tr>
<td>23</td>
<td>11011011</td>
<td>01110100</td>
</tr>
<tr>
<td>24</td>
<td>11011010</td>
<td>01110000</td>
</tr>
<tr>
<td>25</td>
<td>00101010</td>
<td>00101000</td>
</tr>
<tr>
<td>26</td>
<td>01100010</td>
<td>01100001</td>
</tr>
<tr>
<td>27</td>
<td>11011111</td>
<td>01101101</td>
</tr>
<tr>
<td>28</td>
<td>11011110</td>
<td>01101001</td>
</tr>
<tr>
<td>29</td>
<td>00111001</td>
<td>00100100</td>
</tr>
<tr>
<td>30</td>
<td>01100011</td>
<td>01100011</td>
</tr>
<tr>
<td>31</td>
<td>00000111</td>
<td>00000111</td>
</tr>
<tr>
<td>32</td>
<td>00000110</td>
<td>00000110</td>
</tr>
<tr>
<td>33</td>
<td>00011000</td>
<td>00011000</td>
</tr>
<tr>
<td>34</td>
<td>00011001</td>
<td>00011001</td>
</tr>
<tr>
<td>35</td>
<td>00011010</td>
<td>00011010</td>
</tr>
<tr>
<td>36</td>
<td>01100000</td>
<td>01100000</td>
</tr>
<tr>
<td>37</td>
<td>01101101</td>
<td>01101101</td>
</tr>
<tr>
<td>38</td>
<td>00011011</td>
<td>00011011</td>
</tr>
<tr>
<td>39</td>
<td>00101110</td>
<td>00101110</td>
</tr>
<tr>
<td></td>
<td>101 01010</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>-----------</td>
<td>---</td>
</tr>
<tr>
<td>81</td>
<td>110 10100</td>
<td>95</td>
</tr>
<tr>
<td>82</td>
<td>001 00001</td>
<td>96</td>
</tr>
<tr>
<td>83</td>
<td>010 00010</td>
<td>97</td>
</tr>
<tr>
<td>84</td>
<td>10001 101</td>
<td>98</td>
</tr>
<tr>
<td>85</td>
<td>10011 010</td>
<td>99</td>
</tr>
<tr>
<td>86</td>
<td>10110 100</td>
<td>100</td>
</tr>
<tr>
<td>87</td>
<td>11010000</td>
<td>101</td>
</tr>
<tr>
<td>88</td>
<td>01011 101</td>
<td>102</td>
</tr>
<tr>
<td>89</td>
<td>10111 111</td>
<td>103</td>
</tr>
<tr>
<td>90</td>
<td>11110110</td>
<td>104</td>
</tr>
<tr>
<td>91</td>
<td>01100 101</td>
<td>105</td>
</tr>
<tr>
<td>92</td>
<td>11000011</td>
<td>106</td>
</tr>
<tr>
<td>93</td>
<td>00001111</td>
<td>107</td>
</tr>
<tr>
<td>94</td>
<td>00111110</td>
<td>108</td>
</tr>
<tr>
<td>95</td>
<td>00110000</td>
<td>109</td>
</tr>
<tr>
<td>96</td>
<td>11111000</td>
<td>110</td>
</tr>
<tr>
<td>97</td>
<td>11111001</td>
<td>111</td>
</tr>
<tr>
<td>98</td>
<td>01111011</td>
<td>112</td>
</tr>
<tr>
<td>99</td>
<td>11111111</td>
<td>113</td>
</tr>
<tr>
<td>100</td>
<td>01110111</td>
<td>114</td>
</tr>
<tr>
<td>101</td>
<td>11100111</td>
<td>115</td>
</tr>
<tr>
<td>102</td>
<td>01000111</td>
<td>116</td>
</tr>
<tr>
<td>103</td>
<td>10000111</td>
<td>117</td>
</tr>
</tbody>
</table>

It can be observed that all the binary 8-tuples listed above have even weight and along with the all 0 8-tuple, these 8-tuples form a vector space of dimension 7.
APPENDIX B

Weight Distribution of (27, 25) and (28, 25) RS Codes

The number of code words having exactly \( h \) non-zero symbols for a maximum distance separable code is given by Equation (4.2.2). If \( A_1(h) \) and \( A_2(h) \) represent the Hamming weight distribution of (27, 25, 3) and (28, 25, 4) RS codes defined over \( GF(2^7) \) respectively, then Equation (4.2.2) can be used to obtain the following table.

<table>
<thead>
<tr>
<th>( h )</th>
<th>( A_1(h) )</th>
<th>( A_2(h) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>37475</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>2.786 E8</td>
<td>2.600 E06</td>
</tr>
<tr>
<td>5</td>
<td>1.628 E11</td>
<td>1.548 E09</td>
</tr>
<tr>
<td>6</td>
<td>7.581 E13</td>
<td>7.538 E11</td>
</tr>
<tr>
<td>7</td>
<td>2.888 E16</td>
<td>3.001 E14</td>
</tr>
<tr>
<td>8</td>
<td>9.170 E18</td>
<td>1.003 E17</td>
</tr>
<tr>
<td>9</td>
<td>2.459 E21</td>
<td>2.831 E19</td>
</tr>
<tr>
<td>10</td>
<td>5.620 E23</td>
<td>6.330 E22</td>
</tr>
<tr>
<td>11</td>
<td>1.103 E26</td>
<td>1.419 E24</td>
</tr>
<tr>
<td>12</td>
<td>1.868 E28</td>
<td>2.554 E26</td>
</tr>
<tr>
<td>13</td>
<td>2.737 E30</td>
<td>3.992 E28</td>
</tr>
<tr>
<td>14</td>
<td>3.476 E32</td>
<td>5.432 E30</td>
</tr>
<tr>
<td>15</td>
<td>3.826 E34</td>
<td>6.439 E32</td>
</tr>
<tr>
<td>16</td>
<td>3.645 E36</td>
<td>6.644 E34</td>
</tr>
<tr>
<td>17</td>
<td>2.995 E38</td>
<td>5.956 E36</td>
</tr>
<tr>
<td>18</td>
<td>2.113 E40</td>
<td>4.622 E37</td>
</tr>
<tr>
<td>19</td>
<td>1.271 E42</td>
<td>3.090 E40</td>
</tr>
<tr>
<td>20</td>
<td>6.458 E43</td>
<td>1.766 E42</td>
</tr>
<tr>
<td>21</td>
<td>2.734 E45</td>
<td>8.543 E43</td>
</tr>
<tr>
<td>22</td>
<td>9.469 E46</td>
<td>3.452 E45</td>
</tr>
<tr>
<td>23</td>
<td>2.614 E48</td>
<td>1.144 E47</td>
</tr>
<tr>
<td>24</td>
<td>5.533 E49</td>
<td>3.026 E48</td>
</tr>
<tr>
<td>25</td>
<td>8.433 E50</td>
<td>6.149 E49</td>
</tr>
<tr>
<td>26</td>
<td>8.238 E51</td>
<td>9.011 E50</td>
</tr>
<tr>
<td>27</td>
<td>3.875 E52</td>
<td>8.477 E51</td>
</tr>
</tbody>
</table>

where a \( E_b \) means \( a \times 10^b \).

Note that \( \sum_{h=0}^{27} A_1(h) = \sum_{h=0}^{28} A_2(h) = 128^{25} \) as the code is defined on \( GF(2^7) \) and the number of information symbols in each code is 25.