Low Complexity Bit-level Soft-decision Decoding
for Reed-Solomon Codes

by

Min-seok Oh

Thesis submitted to the University of Surrey
for the degree of
Doctor of Philosophy

Centre for Communication Systems Research
University of Surrey
Guildford, Surrey
UK

September 1999

© M. Oh
ABSTRACT

Reed-Solomon codes (RS codes) are an important method for achieving error-correction in communication and storage systems. However, it has proved difficult to find a soft-decision decoding method which has low complexity. Moreover, in some previous soft-decision decoding approaches, bit-level soft-decision information could not be employed fully. Even though RS codes have powerful error correction capability, this is a critical shortcoming.

This thesis presents bit-level soft-decision decoding schemes for RS codes. The aim is to design a low complexity sequential decoding method based on bit-level soft-decision information approaching maximum likelihood performance.

Firstly a trellis decoding scheme which gives easy implementation is introduced, since the soft-decision information can be used directly. In order to allow bit-level soft-decision, a binary equivalent code is introduced and Wolf's method is used to construct the binary-trellis from a systematic parity check matrix.

Secondly, the Fano sequential decoding method is chosen, which is sub-optimal and adaptable to channel conditions. This method does not need a large amount of storage to perform an efficient trellis search. The Fano algorithm is then modified to improve the error correcting performance.

Finally, further methods of complexity reduction are presented without loss of decoding performance, based on reliability-first search decoding using permutation groups for RS codes. Compared with the decoder without permutation, those schemes give a large complexity reduction and performance improvement approaching near-maximum likelihood performance. In this thesis, three types of permutation, cyclic, squaring and hybrid permutation, are presented and the decoding methods using them are implemented.
Acknowledgment

First of all, I thank God who has brought me to this position.

I would like to express my sincere gratitude to my supervisor, Dr. Peter Sweeney. He provided me with invaluable technical advice and constructive criticism throughout my academic years. I will not forget him helping me in the most important period of my life. I wish to thank my colleague, David Burgess, who gave many useful comments.

I would also like to thank my family including my parents, sister, brother-in-law, brother, and sister-in-law. Without their support, I would not have finished this research. I am proud of my family all the time.

I am indebted to the friends who I have made over this period. Especially, I thank Geetag for encouraging and advising me in many things. I will miss the great times which I spent with him.

Finally, I am so happy to have a lovely lady, Meeree, who showed me true love and affection during these otherwise lonely years. She has waited for me with patience and has been the hope of my life. I dedicate this thesis to her.

Min-seok Oh

September 1999
# TABLE OF CONTENTS

ABSTRACT ............................................................................................................................... i

ACKNOWLEDGEMENT ............................................................................................................. ii

1. INTRODUCTION

1.1. Preamble ............................................................................................................................ 1

1.2. Decoding for Reed-Solomon Codes .............................................................................. 2

1.3. Overview of the thesis ..................................................................................................... 5

1.4. Original work ................................................................................................................... 7

2. CODING THEORY FOR RELIABLE COMMUNICATIONS

2.1. Introduction ...................................................................................................................... 8

2.2. Algebra ............................................................................................................................ 9

2.2.1. Groups, Rings and Fields ......................................................................................... 9

2.2.2. Properties of Galois Fields ...................................................................................... 10

2.2.3. Vector Space Representation .................................................................................... 14

2.2.4. Matrices ...................................................................................................................... 15

2.3. Linear Cyclic Codes ........................................................................................................ 16

2.3.1. Linear Block codes .................................................................................................... 16

2.3.2. Cyclic codes ................................................................................................................ 18

2.3.2.1. Properties of cyclic codes .................................................................................. 18

2.3.2.2. Generator and Parity check Matrices of cyclic codes ......................................... 19

2.3.3. BCH and Reed-Solomon Codes .............................................................................. 20

2.3.3.1. BCH codes .......................................................................................................... 20

2.3.3.2. Reed-Solomon Codes ......................................................................................... 21

2.3.3.3. Binary Equivalents of Reed-Solomon codes ...................................................... 22

2.3.3.4. Encoding of Binary Systematic Equivalents of Reed-Solomon Codes .............. 24

2.4. Key Concepts of Coding System .................................................................................. 26

2.4.1. Channel Noise .......................................................................................................... 26

2.4.2. Digital Channel Model ............................................................................................. 27

2.4.3. Error Bit Probability in AWGN .............................................................................. 28

2.4.4. Soft-decision Decoding ............................................................................................ 29

2.4.5. Coding Gain .............................................................................................................. 30
3. SOFT-DECISION DECODING METHODS FOR REED-SOLOMON CODES

3.1. Introduction ...............................................................................................................33

3.2. Algebraic Decoding for Soft-decision .....................................................................34

3.2.1. Forney's Method ..................................................................................................34

3.2.2. Chase Algorithm ................................................................................................35

3.3. Trellis Structure for Block Codes ...................................................................    36

3.3.1. Trellis for Block codes .........................................................................................37

3.3.2. Trellis Complexity ..............................................................................................38

3.3.3. Wolf's Trellis for Reed-Solomon codes ................................................................39

3.4. Trellis Decoding Approaches ..................................................................................42

3.4.1. Viterbi Algorithm ...............................................................................................43

3.4.2. Reduced Search Algorithm ..................................................................................43

3.4.3. Sequential Decoding for Reed-Solomon Codes ...................................................45

3.5. Low complexity Bit-level Soft-decision Sequential Decoding for Reed-Solomon

Codes ................................................................................................................................47

3.5.1. Binary-branch Trellis Structure .........................................................................48

3.5.2. Modification of Fano Algorithm ..........................................................................48

3.5.3. Permutation Decoding ..........................................................................................49

3.6. Simulation Test Bed .................................................................................................50

3.6.1. User Interface ......................................................................................................50

3.6.2. Initializing Mode ..................................................................................................51

3.6.3. Decoding Simulation ...........................................................................................51

3.6.4. Performance Evaluation .......................................................................................52

3.7. Bit Reliability by Soft-decision ...............................................................................52

3.8. Discussion .................................................................................................................55

4. SEQUENTIAL DECODING APPROACH FOR REED-SOLOMON CODES

4.1. Introduction ...............................................................................................................56

4.2. A Survey of Sequential Decoding Algorithm ..........................................................57

4.3. Definition of Metric in Sequential Decoding ..........................................................58

4.4. Complexity Problem of Sequential Decoding ........................................................60

4.5. Sequential Decoding for Block Codes ....................................................................62

4.5.1. Stack Algorithm ................................................................................................62

4.5.2. Fano Algorithm .................................................................................................63
6.2.4. Simulation Results in SQPSD .......................................................... 111
  6.2.4.1. Error Location Distribution by Squaring Permutation .......... 112
  6.2.4.2. Performance Analysis of SQPSD ........................................... 114
6.2.5. Discussion .................................................................................. 116

6.3. Hybrid Permutation Sequential Decoding ...................................... 116
  6.3.1. Hybrid permutation of Reed-Solomon codes ......................... 117
  6.3.2. Hybrid Permutation Sequential Decoding (HPSD) ................. 118
  6.3.3. Simulation Results in HPSD .................................................... 120
    6.3.3.1. Error Location Distribution by Hybrid Permutation ........ 121
    6.3.3.2. Performance Analysis in HPSD ........................................ 122
      6.3.3.2.1. Decoding Performance according to maximum number of trial sequences Y 123
      6.3.3.2.2. Cost-effective Performance in HPSD ....................... 124
  6.3.4. Discussion .............................................................................. 127

6.4. Performance Evaluation .............................................................. 127
  6.4.1. Decoding Performance Analysis .............................................. 128
  6.4.2. Complexity Analysis ............................................................. 132
    6.4.2.1. Computational Distribution according to Permutation Methods 132
    6.4.2.2. Complexity Comparison according to Permutation Decoding Methods 134
  6.4.3. Discussion .............................................................................. 135

7. CONCLUDING REMARKS AND FURTHER WORK
  7.1. Introduction ............................................................................. 136
  7.2. Conclusion .............................................................................. 136
  7.3. Further Work .......................................................................... 139
    7.3.1. Shortened Reed-Solomon codes using HPSD ..................... 140
    7.3.2. Subcode of Reed-Solomon codes using HPSD .................... 141
    7.3.3. Concatenated Coding Scheme using HPSD ....................... 142

APPENDIX
  APPENDIX A. GENERATOR POLYNOMIAL FOR REED-SOLOMON CODES . 144
  APPENDIX B. GALOIS FIELD REPRESENTATION FOR REED-SOLOMON CODES 145
  APPENDIX C. WEIGHT DISTRIBUTION FOR REED-SOLOMON CODES .... 147
  APPENDIX D. ALGORITHM FOR SYSTEMATIC MATRIX ....................... 151

REFERENCES .................................................................................. 152
CHAPTER 1

INTRODUCTION

1.1 PREAMBLE

The basic idea of reliable communication was introduced by Claude Shannon in a famous paper\cite{51} in 1948. In that paper, he derived a concept of channel capacity, which gives an upper limit for the rate of information transmission and he showed that sufficiently long coded transmitted signals can produce arbitrarily low error probability in the delivered information.

Error control coding was introduced as an important method for error detection and correction by the addition of redundancy to transmitted information data. There has been a lot of progress in this area with discoveries of promising code classes and decoding algorithms. Richard Hamming\cite{23} introduced error correcting codes and a method for error detection in 1950 and many other codes and decoding methods were developed during the 1950s and 1960s. Amongst them, an important family of non-binary codes was discovered by I.S. Reed and G. Solomon in 1960: Reed-Solomon codes (RS codes).

An important feature of RS codes is that they are based on the arithmetic of finite fields and consist of groups of bits. This feature is very powerful for correcting burst errors, since several error bits can be regarded as a small number of error events. Burst errors may happen in radio channels as a result of signal fading, in wire and cable transmission affected by impulse switching noise and crosstalk, and in magnetic recording which is subject to tape dropouts due to surface defects and dust particles. Thus RS codes have been amongst the most important codes for use in various practical applications including storage devices, satellite communications, and deep-space communications. The development of an efficient decoding method for RS codes will always be of interest. In this thesis, a new decoding method is presented.
1.2 DECODING FOR REED-SOLOMON CODES

Many early approaches for decoding of RS codes were based on their algebraic background (see Gorenstein and Zierler[22], Massey[36], and Sugiyama[53]). Although these offered many useful algorithms and implementation methods, channel measurement information (or soft-decision) could not be employed in their work.

As is well explained in many works[11][38], soft-decision decoding methods produce the same decoding error probability at \( \frac{1}{2} \) the transmitted power per information bit of hard-decision decoding methods. Thus finding soft-decision decoding methods for RS codes is an important issue in the coding area.

Chase[9] and Forney[18] introduced interesting methods for soft-decision decoding of block codes, which can be applied to RS codes. Their algorithms illustrated the trade-off between complexity and decoding performance. Chase’s algorithm shows a good decoding performance but it gives rise to a complexity problem. On the other hand Forney’s algorithm has reasonable complexity but the decoding performance is not so good even though some improvement has been presented by Taipale and Pursley[54]. However, these two algorithms would not be suitable for long RS codes for which high computations are required due to a large number of code words.

A general coset-decoding principle for maximum likelihood decoding of block and lattice codes is described in[12]. However, it was implemented[16][12][13] with respect to subcodes that belong to a limited class. Properties of the coset representation of the code are employed in[16] more intensively than in [12][13] and in a different manner. The methods developed in [16][12][13] are not efficient in the general case although they are useful for several specific codes. In particular they may be inefficient for longer codes with reasonably high rates where the dimension of the largest available subspace of the two types utilized is a small fraction of the dimension of the codes, and the numerous coset representatives lack an attractive structure. Thus this decoding approach does not seem suitable for long RS codes.
As another soft-decision decoding approach, the use of a trellis seems attractive, since soft-decision values can be easily employed. Originally the trellis was introduced for convolutional codes by Forney[20]. In 1974, Bahl, Cocke, Jelinek, and Raviv[3] presented a method of representing the code words in any linear block code by the path labels in a trellis. In 1978, Wolf[59] demonstrated maximum likelihood decoding of any linear block codes using soft-decision. Thus it is possible to implement trellis decoding for RS codes. The basic algorithm of trellis decoding finds the most likely code word given the received sequence, using the log-likelihood function associated with soft-decision values. Thus an important task of trellis decoding has been to look for an efficient trellis search algorithm to find the maximum likelihood(ML) code word.

The Viterbi algorithm[57] is an optimal method for trellis decoding assuming equally likely sequences and a memoryless channel. However, this algorithm cannot feasibly be implemented in codes with constraint length over about 10 because the complexity increases exponentially with the length. Moreover it is not adaptable to channel conditions, performing the same number of computations regardless of noise level.

Matis[37] introduced a reduced-search soft-decision trellis decoding algorithm, which can be regarded as a generalization of the Viterbi algorithm. However, this approach was not able to achieve sufficiently high performance compared to an optimal decoder such as the Viterbi algorithm. A desirable decoder will be cost-effective considering both complexity and decoding performance.

Sequential decoding has been considered as a good alternative to the Viterbi algorithm. The great advantage of sequential decoding over Viterbi decoding is that its complexity does not increase exponentially with the code length. There was an interesting approach by Shin[52] based on a modified Fano sequential algorithm. In this approach, many good results were achieved but it also had at least two drawbacks.

The first of those is another complexity problem arising from non-binary branches of RS codes. This may be much more serious if the symbol field is large. The other drawback is that the approach does not employ bit level soft-decision information to
1.2 Decoding for Reed-Solomon Codes

the same extent as the previous methods. Concerning that shortcoming, Berlekamp et al.[7] in 1987 stated that "The major drawback with RS codes (for satellite use) is that the present generation of decoders does not make full use of bit-level soft-decision information". Although over a decade has passed since that statement, only limited progress[56] has been made in soft decision decodings for RS codes.

In this thesis, some bit-level soft decision decoding schemes for RS codes are presented with near-ML performance with reasonable complexity. For the implementation of this decoding, we construct a binary trellis, which is based on Wolf's method[59].

As a trellis search algorithm, the Fano algorithm is modified so as to improve the decoding performance. Furthermore, permutation techniques are introduced to reduce the complexity without any performance degradation.

Permutation groups are found by the mathematical properties of RS codes. We present three kinds of permutation techniques, cyclic, squaring, and hybrid permutation. These permutation groups are used to give a sequence which is expected to involve the lowest computation cost.

Chapter 1. Introduction
1.3 OVERVIEW OF THE THESIS

A brief description of each chapter in the thesis follows.

Chapter 2 is a review of coding theory including the mathematical background. In this chapter, basic algebraic properties of finite fields are introduced and some interesting block codes are explained. There is also an introduction of some important concepts in the application of coding to the area of communications.

Chapter 3 presents trellis decoding and trellis structure for block codes. In this chapter, the trellis for block codes based on Wolf's method is studied and trellis complexity is discussed generally. The binary trellis structure for RS codes allowing bit-level soft-decision information to be employed is also explained. Other approaches for soft-decision decoding, the GMD algorithm, the Chase algorithm, reduced search soft-decision trellis decoding and sequential decoding, are reviewed. The trade-off between complexity and decoding performance is discussed for each of those methods.

Chapter 4 is a discussion of sequential decoding. In this chapter there is a survey for the purpose of determining the most efficient sequential algorithm for RS codes. The computational characteristics of sequential decoding are examined in the case of the Fano sequential decoding algorithm. In addition three types of decoding error events are specified and a modification of the Fano algorithm which achieves near-ML decoding performance is explained.

In Chapter 5, a low complexity decoding method for RS codes, using permutation techniques, is introduced. First of all, we explain the concepts for reliability-first searching sequential decoding. Then the cyclic permutation technique is discussed for the design of a low complexity sequential decoder. Finally, through simulation, performance and complexity are compared with non-permutation decoding.

In Chapter 6, we present an advanced permutation decoding method for further complexity reduction. Firstly, we introduce a squaring permutation technique which
gives a different set of sequences compared with cyclic permutation. A squaring permutation decoder is described with an analysis of decoding performance and complexity. Secondly, we introduce another permutation technique, hybrid permutation, for a modified sequential decoder. This technique combines the squaring and cyclic permutation techniques to maximize the searching efficiency of the decoder. Finally, there is an evaluation of the permutation techniques used in terms of performance and complexity.

Chapter 7 is devoted to the conclusions and further work. We explain the trade-offs between the many decoding methods which have been employed in the thesis. The decoding method which is considered the most cost-effective is indicated. Finally some further work is suggested.
1.4 ORIGINAL WORK

The original contributions in this thesis are as follows.

- Low complexity bit-level soft-decision sequential decoding for RS codes
- Modified Fano algorithm for ML decoding
- Low complexity sequential decoder using cyclic permutation
- Low complexity sequential decoder using squaring permutation on normal basis
- Hybrid-permutation decoding using cyclic and squaring permutation
- Criteria to decide the complexity level of sequences used in sequential decoding
- Criteria for choice of effective decoding strategy for RS codes based on reliability-first search

- Publications and presentations include:
CHAPTER 2

CODING THEORY FOR RELIABLE COMMUNICATIONS

2.1. INTRODUCTION

In the progress of coding theory, mathematical structure has been the basis for defining a given code. Thus it is necessary to study basic algebraic structure for understanding coding theory. In this chapter, the basic mathematical background and its application to coding theory are reviewed.

Firstly, the properties of groups, finite fields, vector spaces, and matrices are studied. On those properties, the basic theory of linear cyclic codes is developed and then, as an important class of linear cyclic codes, BCH and RS codes are defined. Next we introduce a binary equivalent of the RS code, which is used to implement bit-level soft-decision decoding that is a key subject in this thesis.

The remainder of this chapter is devoted to describing the fundamental concepts in a communication system which uses coding. There is a description of the basic channel model and bit error probability is discussed in a particular channel model. Then soft-decision decoding is described and maximum likelihood decoding is defined in the soft-decision decoding. Finally we describe the coding gain which is an important concept in coding. This is used to specify the improvement achieved by a coding system. We also verify that soft-decision decoding produces about 3.0 dB gain over hard-decision on the AWGN(Additive White Gaussian Noise) channel.
2.2. ALGEBRA

Certain concepts from algebra are essential for understanding coding theory. In this section, we study important concepts which will be used in later chapters. Much of the contents are taken from textbooks by Lin and Costello [31], Wicker [58], Macwilliams and Sloane [33], Clark and Cain [11], and Micheson and Levesque [38] where proofs of the theorems can also be found.

2.2.1 GROUPS, RINGS AND FIELDS

Definition 2.1: A group $G$ is a set of elements for which an operation "$\circ$" is defined and for which the following conditions are satisfied:

(i) **Closure:** For any $a$ and $b$ in $G$, there exists an element $c$ in $G$ satisfying $a \circ b = c$.

(ii) **Associative:** For $a$, $b$, and $c$ in $G$, $(a \circ b) \circ c = a \circ (b \circ c)$.

(iii) **Identity:** $G$ contains an element $e$ such that, for any $a$ in $G$, $a \circ e = e \circ a = a$. This element $e$ is called an identity element of $G$.

(iv) **Inverse:** For any element $a$ in $G$, there exists another element $a'$ in $G$ such that $a \circ a' = a' \circ a = e$. This element $a'$ is called an inverse of $a$ ($a$ is also an inverse of $a'$).

A group is said to be commutative if it also satisfies

**Commutativity:** For all $a, b \in G$, $a \circ b = b \circ a$.

Definition 2.2: A ring $R$ is a set of elements for which two operations, addition $\oplus$ and multiplication $\cdot$, are defined and for which the following conditions are satisfied:

(i) $R$ is a commutative group under addition $\oplus$.

(ii) **Associative Law:** For any element $a$, $b$, and $c$ in $R$, $(a \cdot b) \cdot c = a \cdot (b \cdot c)$.

(iii) **Distributive Law:** For any three elements $a$, $b$, and $c$ in $R$, $a \cdot (b \oplus c) = (a \cdot b) \oplus (a \cdot c)$ and $(b \oplus c) \cdot a = (b \cdot a) \oplus (c \cdot a)$.

Definition 2.3: A field $F$ is defined as ring with a multiplicative identity in which every nonzero element has a multiplicative inverse. The number of elements in a field
is defined as the order of the field. Fields of finite order (cardinality) are also called Galois fields in honour of the discoverer, Evariste Galois. A Galois field of order \( q \) is denoted by \( GF(q) \).

### 2.2.2 PROPERTIES OF GALOIS FIELDS

Definition 2.4: The characteristic of a Galois Field is defined as the smallest positive integer \( \lambda \) such that \( \sum_{i=1}^{\lambda} 1 = 0 \).

Theorem 2.1: The characteristic \( \lambda \) of a Galois field \( GF(q) \) is prime.

Definition 2.5: Let \( \alpha \) be an element of \( GF(q) \). The order of \( \alpha \) is defined as the smallest positive integer \( t \) such that \( \alpha^t = 1 \).

Theorem 2.2: Let \( t \) be the order of \( \alpha \) for \( \alpha \in GF(q) \). Then \( t \) divides \( q-1 \).

Definition 2.6: A nonzero element \( \alpha \) with order \( (q-1) \) is defined as a primitive element in \( GF(q) \). Therefore all nonzero elements in \( GF(q) \) can be represented as \( (q-1) \) consecutive powers of a primitive element \( \alpha \) (Power Representation of Galois Fields).

Definition 2.7: Let a polynomial \( f(x) \) with variable \( x \) and coefficients \( a_i \) be of the following form

\[
f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 \quad (2.1)
\]

where \( a_i \in GF(q) \) for \( 0 \leq i \leq n \). The degree of a polynomial \( f(x) \) is the largest power of \( x \) with nonzero coefficient. \( GF(q)[x] \) denotes the collection of all \( f(x) \) of any degree. Then a polynomial \( f(x) \) is irreducible in \( GF(q) \) if \( f(x) \) cannot be divided by a lower-degree polynomial in \( GF(q)[x] \).

---

Chapter 2. Coding Theory for Reliable Communications
Definition 2.8: An irreducible polynomial \( p(x) \in GF(q)[x] \) of degree \( m \) is said to be **primitive polynomial** if the smallest positive integer \( n \) for which \( p(x) \) divides \( x^n - 1 \) is \( n = q^m - 1 \).

Definition 2.9: Let \( \alpha \) be an element in the field \( GF(q^m) \). The **minimal polynomial** of \( \alpha \) with respect to \( GF(q) \) is the smallest degree nonzero polynomial \( p(x) \) in \( GF(q)[x] \) such that \( p(\alpha) = 0 \).

Theorem 2.3: If \( GF(q^m) \) has been constructed using an \( m \)-th degree primitive polynomial \( p(x) \), the field elements can be represented by polynomial of the form as

\[
a_0 + a_1\alpha + \ldots + a_{m-1}\alpha^{m-1}
\]

where \( \alpha \) is a root of \( p(x) \) and \( a_i \in GF(q) \).

Definition 2.10: Let \( V \) be a set of elements for field \( F \). The set \( V \) is defined as a vector space over field \( F \) if the following conditions are satisfied:

(i) \( V \) is a commutative group under addition

(ii) For any element \( a \) of \( F \) and any element \( v \) of \( V \), \( a \cdot v \) is an element of \( V \).

For any element \( u \) and \( v \) of \( V \) and any elements \( a \) and \( b \) of \( F \),

(iii) \( a \cdot (u + v) = a \cdot u + a \cdot v \),

(iv) \( (a + b) \cdot v = a \cdot v + b \cdot v \)

(v) \( (a \cdot b) \cdot v = a \cdot (b \cdot v) \).

(vi) For any \( v \) in \( V \) and any \( a \) and \( b \) in \( F \), let \( 1 \) be the multiplicative identity of \( F \). Then, for any \( v \) in \( V \), \( 1 \cdot v = v \).

The elements of \( V \) are called **vectors** and the elements of the field \( F \) are **scalars**.

Definition 2.11: A **basis** of the vector space over \( GF(2^m) \) is defined as a set of \( m \) linearly independent elements \( \lambda_0, \lambda_1, \ldots, \lambda_{m-1} \) in \( GF(2^m) \). Any element \( \alpha \) over \( GF(2^m) \) can be represented by

\[
\alpha = b_0\lambda_0 + b_1\lambda_1 + \ldots + b_{m-1}\lambda_{m-1}
\quad \text{for } b_i \in GF(2) .
\]
Definition 2.12: If $GF(p^m)$ is a vector space of $GF(p)$, the sum

$$tr(\beta) = \beta + \beta^p + \beta^{p^2} + \cdots + \beta^{p^{m-1}} = \sum_{j=0}^{m-1} \beta^{p^j}$$

(2.4)

is called the trace of $\beta \in GF(p^m)$.

Definition 2.13: If $\alpha$ is a root of a primitive polynomial over $GF(2)$, the polynomial basis over $GF(2^m)$ is defined as the set \( \{ \lambda_0, \lambda_1, \cdots, \lambda_{m-1} \} \) such that $\lambda_i = \alpha^i$. Any symbol $\beta$ is represented by

$$\beta = a_0 + a_1 \alpha + \cdots + a_{m-1} \alpha^{m-1}$$

(2.5)

Definition 2.14: The normal basis over $GF(2^m)$ is defined as a set of linearly independent roots over $GF(2)$ with the form $\{ \lambda_0, \lambda_1, \cdots, \lambda_{m-1} \}$ where $\lambda_i = \beta^i$ for $\beta \in GF(2^m)$. Perlis[44] has shown that a necessary and sufficient condition for a normal basis is

$$tr(\beta) = \sum_{j=0}^{m-1} \beta^{p^j} = 1.$$ 

(2.6)

Definition 2.15: For a prime number $p$, $GF(p^m)$ can be viewed as a construction over $GF(p)$. Fields of order $p^m$ are called extension of $GF(p)$. The subfields of $GF(p^m)$ are defined as $GF(p^b)$ where $b|m$ (“$b$ divides $m$”).

Example:

<table>
<thead>
<tr>
<th>Galois Fields</th>
<th>Subfields</th>
</tr>
</thead>
<tbody>
<tr>
<td>$GF(2^3)$</td>
<td>$GF(2), GF(2^3)$</td>
</tr>
<tr>
<td>$GF(2^4)$</td>
<td>$GF(2), GF(2^2), GF(2^4)$</td>
</tr>
<tr>
<td>$GF(2^5)$</td>
<td>$GF(2), GF(2^5)$</td>
</tr>
<tr>
<td>$GF(2^6)$</td>
<td>$GF(2), GF(2^2), GF(2^3), GF(2^6)$</td>
</tr>
</tbody>
</table>

Chapter 2. Coding Theory for Reliable Communications
Definition 2.16: For $\beta \in GF(q^n)$, the \textit{conjugates} of $\beta$ with respect to the subfield $GF(q)$ are the elements $\beta^i$ for $i = 1, 2, \ldots$. The set of conjugates is called as the \textit{conjugacy class}.

Theorem 2.4: The conjugacy class of $\alpha \in GF(q^n)$ with respect to $GF(q)$ contains $d$ elements, where $\alpha^{d^r} = \alpha$ and $d \mid m$.

Definition 2.17: The \textit{cyclotomic cosets modulo $n$} with respect to $GF(q)$ are a partitioning of the integers $\{0, 1, \ldots, n - 1\}$ into sets of the form

$$C_i = \{s, qs, q^2s, q^3s, \ldots, q^{m_i} s\}$$

where $m_i$ is the smallest positive integer such that $(q^m \cdot s) \mod n = s$.

Example: The cyclotomic cosets mod 15 (with $q = 2$) are:

$$C_0 = \{0\},
C_1 = \{1, 2, 4, 8\},
C_3 = \{3, 6, 12, 9\},
C_5 = \{5, 10\},
C_7 = \{7, 14, 13, 11\}.$$

Definition 2.18: For $\alpha \in GF(q^n)$, the \textit{minimal polynomial} of $\alpha$ with respect to $GF(q)$ is the smallest degree nonzero polynomial $p(x)$ in $GF(q)[x]$ such that $p(\alpha) = 0$.

Theorem 2.5: Let $\beta$ be an element in an extension field of $GF(2)$. For $f(X) \in GF(2)[X]$, if $\beta$ is a root of $f(X)$, then for any $t \geq 0$, $\beta^{2^t}$ is also a root of $f(X)$.

Definition 2.19: Automorphisms of $GF(p^n)$ are the mappings of the field which fix every element of the subfield $GF(p)$ and preserve addition and multiplication. An \textit{automorphism} of $GF(p^n)$ over $GF(p)$ is a mapping $\sigma: \beta \rightarrow \beta^\sigma$, which fixes the elements of $GF(p)$ and has the properties

(i) $(\alpha + \beta)^\sigma = \alpha^\sigma + \beta^\sigma$
(ii) $(\alpha \beta)^\sigma = \alpha^\sigma \beta^\sigma$. 

\hfill Chapter 2. Coding Theory for Reliable Communications
The set of all automorphisms of \( GF(p^m) \) forms a group if we define the product of \( \sigma \) and \( \tau \) by

\[
\sigma \circ \tau = (\sigma^\tau)^\circ.
\]

This group is called the automorphism group of \( GF(p^m) \).

Theorem 2.6: The automorphism group of \( GF(p^m) \) is the cyclic group of order \( m \) consisting of the mapping \( \sigma_p: \beta \to \beta^p \) and its powers.

### 2.2.3 VECTOR SPACE REPRESENTATION

Definition 2.20: Let \( S \) be a non-empty subset of a vector space \( V \) over a field \( F \). Then \( S \) is defined as a subspace if the following conditions are satisfied:

- For any two vectors \( u \) and \( v \) in \( S \), \( u + v \) is also a vector in \( S \).
- For any element \( a \) in \( F \) and any vector \( u \) in \( S \), \( a \cdot u \) is also in \( S \).

Definition 2.21: Let \( (v_1, v_2, \ldots, v_k) \) be \( k \) vectors in a vector space \( V \) over a field \( F \). Let \( (a_1, a_2, \ldots, a_k) \) be scalars from \( F \). Then

\[
a_1v_1 + a_2v_2 + \cdots + a_kv_k
\]

is a linear combination of the vectors \( (v_1, v_2, \ldots, v_k) \).

Definition 2.22: A set of vectors of \( (v_1, v_2, \ldots, v_k) \) is linearly independent if and only if there are no scalars \( (c_1, c_2, \ldots, c_k) \), not all zero, such that

\[
c_1v_1 + c_2v_2 + \cdots + c_kv_k = 0
\]

Definition 2.23: If every vector in a vector space \( V \) is a linear combination of the vectors in a set, the set of the vectors is said to span the vector space.
Theorem 2.7: If a set of \( k \) vectors \( (v_1, v_2, \ldots, v_k) \) spans a vector space that contains a set of \( m \) linearly independent vectors \( u_1, u_2, \ldots, u_m \), then \( k \geq m \).

Theorem 2.8: If two sets of linearly independent vectors span the same space, there are the same number of vectors in each set.

Definition 2.24: A set of \( k \) linearly independent vectors spanning a \( k \)-dimensional vector space is called a basis of the space.

Theorem 2.9: If \( V \) is a \( k \)-dimensional vector space, any set of \( k \) linearly independent vectors in \( V \) is a basis for \( V \).

### 2.2.4 MATRICES

Matrices are useful to represent a vector space for coding theory. An \( n \times m \) matrix is an ordered set of \( nm \) elements with a rectangular array of \( n \) rows and \( m \) columns:

\[
\begin{bmatrix}
  a_{11} & a_{12} & \cdots & a_{1m} \\
  a_{21} & a_{22} & \cdots & a_{2m} \\
  \vdots & \vdots & \ddots & \vdots \\
  a_{n1} & a_{n2} & \cdots & a_{nm}
\end{bmatrix}
\]

(2.9)

In coding theory the elements of a matrix usually represent elements in a field. The \( n \) rows and \( m \) columns correspond to \( n \)-vector (\( n \)-tuple) and \( m \)-vector (\( m \)-tuple).

A set of elementary row operations is defined as:

- Interchange of two rows.
- Multiplication of any row by a nonzero field element.
- Addition of any multiple of one row to another.

The row space is the set of all linear combinations of row vectors and the dimension of the row space is called the row rank. Similarly, the column space consists of the set of all linear combination of column vectors with the dimension of column rank. In the special case when the row rank is equal to column rank, the rank is called the rank of the matrix. In addition the set of elements \( a_{ij} \) for \( i = j \) is called the main diagonal and
the $k \times k$ matrix having all 1's on the main diagonal with all other elements zero is called the identity matrix, $I_k$.

Two matrices can be added if they have the same number of rows and the same number of columns. For $k \times n$ matrices $A = [a_{ij}]$ and $B = [b_{ij}]$, addition of two matrices is done on their corresponding entries as follows:

$$A + B = [a_{ij}] + [b_{ij}] = [a_{ij} + b_{ij}]. \quad (2.10)$$

Multiplication of matrices is obtained only if the number of columns in the first matrix is equal to the number of rows in the second matrix such that

$$C = A \times B = [c_{ij}] \quad (2.11)$$

where $c_{ij} = \sum_{r=0}^{n-1} a_{ir}b_{rj}$.

Let $M$ be a $k \times n$ matrix. The transpose of $M$ is defined as the $n \times k$ matrix whose rows are the columns of $M$ and whose columns are the rows of $M$. A submatrix of a matrix $M$ is a matrix that is obtained by striking out given rows and columns of $M$.

### 2.3. LINEAR CYCLIC BLOCK CODES

Linear cyclic codes are an important class of block codes. In this section, the properties of linear cyclic codes are discussed and BCH and RS codes are introduced. The proofs of theorems of this section are well explained in many textbooks[11][31][38][45].

#### 2.3.1 LINEAR BLOCK CODES

The set of all $n$-tuples with elements over $GF(q)$ is a vector space. A set of these vectors of length $n$ is a linear block code if and only if it is a subspace of an $n$-tuple vector space. Any set of basis vectors for a linear block code $V$ can be represented by the rows of a matrix $G$, called a generator matrix. That is, a code vector is a linear combination of the rows of $G$. If the dimension of the vector space $V$ is $k$, the number
of rows in $G$ is $k$. Thus each different linear combination gives a distinct code vector and there are $q^k$ code vectors. Such a code is called an $(n, k)$ code. Therefore the code vector $v$ can be expressed by the matrix form

$$v = u \cdot G$$

(2.12)

$$= (u_0, u_1, \cdots, u_{k-1}) \cdot \begin{bmatrix} g_0 \\ g_1 \\ \vdots \\ g_{k-1} \end{bmatrix}$$

$$= u_0 g_0 + u_1 g_1 + \cdots + u_{k-1} g_{k-1}$$

where $g_i$ is the row space and $u$ is the information message. If $V$ is a subspace of dimension $k$, its null space is a vector space $P$ with dimension $n - k$. A vector $v$ is in $V$ if and only if

$$v \cdot P^T = 0.$$ 

(2.13)

If $v = (a_0, a_1, a_2, \cdots, a_{n-1})$ and $h_{ij}$ denotes the elements in the $i$-th row and the $j$-th column of a matrix $H$ corresponding to $P$, then (2.13) is also expressed by

$$\sum_{i=0}^{n-1} \sum_{j=0}^{n-k-1} a_i h_{ij} = 0$$

(2.14)

This matrix $H$ is called a parity-check matrix of $V$ and it has dimensions of $(n - k) \times n$.

Definition 2.25: If one matrix can be obtained from another by a combination of row operations and column permutations, the two matrices are equivalent.

If the first $k$ components of the code vector are chosen from information symbols, and each of the last $n-k$ components is a linear combination of the first $k$ components, encoding and decoding can be simplified to get the information symbol. A code of this type is called a systematic code. In the code, the generator matrix $G$ has the form

$$G = [I_k \ P]$$

(2.15)
where $I_k$ is a $k \times k$ identity matrix and $P$ is a $k \times (n-k)$ matrix.

Theorem 2.10: Every linear block code can be expressed as a systematic code.

Theorem 2.11: For a systematic code having the generator matrix in the form of (2.15), the parity check matrix of $H$ is $H = [-P^T I_{n-k}]$ where $I_{n-k}$ is an $(n-k) \times (n-k)$ identity matrix and $P^T$ is the transpose of $P$.

### 2.3.2 CYCLIC CODES

Cyclic codes are very useful codes in the class of linear codes. In general, in order to develop the algebraic properties of a cyclic code, we can use the following polynomial form in which the components of a code vector $v = (v_0, v_1, \cdots, v_{n-1})$ are coefficients:

$$v(X) = v_0 + v_1 X + v_2 X^2 + \cdots + v_{n-1} X^{n-1} = \sum_{i=0}^{n-1} v_i X^i \quad (2.16)$$

#### 2.3.2.1 Properties of Cyclic Codes

Definition 2.26: An $(n, k)$ linear code $C$ is called a cyclic code if every cyclic shift of a code vector is also a code vector.

Theorem 2.12: Let $g(X) = 1 + g_1 X + g_2 X^2 + \cdots + g_{n-1} X^{n-1} + X^n$ be a non-zero code polynomial of minimum degree in an $(n, k)$ cyclic code $C$. A polynomial of degree $(n-1)$ or less is a code polynomial if and only if it is a multiple of $g(X)$.

Theorem 2.13: In an $(n, k)$ cyclic code, there exists one and only one code polynomial of degree $n-k$,

$$g(X) = 1 + g_1 X + g_2 X^2 + \cdots + g_{n-k-1} X^{n-k-1} + g_{n-k} X^{n-k} \quad (2.17)$$
Every code polynomial is a multiple of \( g(X) \) and in an \((n, k)\) cyclic code can be expressed by

\[
\nu(X) = u(X)g(X) = (u_0 + u_1 X + \cdots + u_{k-1} X^{k-1})g(X)
\]  

(2.18)

Theorem 2.14: If a polynomial \( g(X) \) has degree \((n - k)\) and is a factor of \( X^n + 1 \), then \( g(X) \) generates an \((n, k)\) cyclic code. The polynomial \( g(X) \) is called a generator polynomial for the \((n, k)\) cyclic code.

Definition 2.27: Let a polynomial of a cyclic code be \( E(X) \). Then \( E(X) \) is an idempotent if

\[
E(X) = E(X)^2 = E(X^2)
\]  

(2.19)

Theorem 2.15: A cyclic code contains a unique idempotent \( E(X) \).

Definition 2.28: Let \( C \) be a binary code of length \( n \). A permutation of the \( n \) coordinate places changes \( C \) into an equivalent code having many of the same properties as \( C \)(minimum weight, weight distribution, etc.).

Definition 2.29: The permutations of coordinate places which send \( C \) into itself, form the automorphism group of \( C \), denoted by \( \text{Aut}(C) \).

### 2.3.2.2 Generator and Parity Check Matrices of Cyclic Codes

By Theorem (2.13) and Theorem (2.14), we can see that if \( k \) \( n \)-tuples corresponding to the \( k \) code polynomials \( g(X), Xg(X), \ldots, X^{k-1}g(X) \) are used as the rows of a \( k \times n \) matrix, we define the following generator matrix \( G \) of the cyclic codes

\[
G = \begin{bmatrix}
g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 \\
o & g_0 & g_1 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\
0 & 0 & g_0 & g_1 & \cdots & g_{n-k} & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \ddots & \vdots & \ddots & \vdots & \ddots & \vdots \\
0 & 0 & 0 & 0 & g_0 & g_1 & \cdots & g_{n-k}
\end{bmatrix}
\]  

(2.20)

By Theorem 2.14, since the generator polynomial \( g(X) \) is a factor of \( X^n + 1 \), we get a polynomial \( h(x) \) as follows
\[ X^n + 1 = g(X)h(X) \]  \hspace{1cm} (2.21)

where \( h(X) = h_0 + h_1X + \cdots + h_kX^k \). Let \( v(X) = u(X)g(X) \) be a code vector. Multiplying \( v(X) \) by \( h(X) \), we obtain

\[
v(X)h(X) = u(X)g(X)h(X) = u(X)(X^n + 1) = u(X) + X^n u(X). \hspace{1cm} (2.22)
\]

Since \( u(X) \) has degree less than \( k \), the coefficients of \( X^k, X^{k+1}, \ldots, X^{n-1} \) in this expression (2.22) must be zero. Thus \( n-k \) equations are obtained

\[
\sum_{i=0}^{k} h_i v_{n-i-j} = 0 \hspace{1cm} \text{for} \hspace{0.5cm} 1 \leq j \leq n-k. \hspace{1cm} (2.23)
\]

If we take the reciprocal of \( h(X) \)

\[
X^k h(X^{-1}) = h_k + h_{k-1}X + \cdots + h_0 X^k, \hspace{1cm} (2.24)
\]

since \( X^k h(X^{-1}) \) is also a factor of \( X^n + 1 \), it generates an \((n, n-k)\) cyclic code with the following \((n-k) \times n\) matrix as a generator matrix.

\[
H = \begin{bmatrix}
h_k & h_{k-1} & \cdots & \cdots & h_0 & 0 & 0 & 0 & \cdots & 0 \\
0 & h_k & h_{k-1} & \cdots & \cdots & h_0 & 0 & 0 & \cdots & 0 \\
0 & 0 & h_k & h_{k-1} & \cdots & h_0 & 0 & 0 & \cdots & 0 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \cdots & \cdots & h_0 
\end{bmatrix} \hspace{1cm} (2.25)
\]

This matrix \( H \) is called the parity check matrix of the cyclic code \( C \).

### 2.3.3 BCH AND RS CODES

#### 2.3.3.1 BCH Codes

BCH codes are a subclass of cyclic codes for multiple-error correction. The generator polynomial \( g(X) \) of the \( t \)-error-correcting BCH code is the lowest-degree polynomial
over \( GF(2) \) which has \( \alpha, \alpha^2, \ldots, \alpha^{2t} \) as its roots where \( \alpha \) is a primitive element in \( GF(2^m) \). It is represented by

\[
g(X) = \prod_{i=0}^{2t} (x - \alpha^i) \quad \text{for } m_0 = 1. \quad (2.26)
\]

Let \( \phi_i(X) \) be the minimal polynomial of \( \alpha^i \). Since the conjugate of each \( \alpha^i \) is also a root of \( g(X) \) by Theorem 2.5, \( g(X) \) must be the least common multiple of \( \phi_1(X), \phi_2(X), \ldots, \phi_{2t}(X) \), that is,

\[
g(X) = LCM\{\phi_1(X), \phi_2(X), \ldots, \phi_{2t}(X)\}.
\]

Since an even power of \( \alpha \) has the same minimal polynomial as an odd power of \( \alpha \), the expression of \( g(X) \) is reduced to

\[
g(X) = LCM\{\phi(X), \phi_3(X), \ldots, \phi_{2t}(X)\}. \quad (2.27)
\]

Thus a binary BCH code with design error correction capability \( t' \) has the parameters as shown in Table 2.1.

<table>
<thead>
<tr>
<th>Table 2.1 Parameters of BCH Codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block Length</td>
</tr>
<tr>
<td>Number of parity check digits</td>
</tr>
<tr>
<td>Minimum distance</td>
</tr>
</tbody>
</table>

### 2.3.3.2 Reed-Solomon Code (RS code)

The binary BCH codes can be generalized to non-binary codes. If \( q \) is a prime number, there are \( q^m \)-ary codes with symbols from \( GF(q) \) for \( m \geq 3 \). An \( (n, k) \) linear code with symbols from \( GF(q^m) \) has a \( k \)-dimensional subspace over \( GF(q^n) \) as its information block within the \( n \)-tuples. For any choice of positive integer \( s \) and \( t \), there exists a \( q^m \)-ary BCH code of length \( n = (q^m)^s - 1 \) for correction of \( t \) or fewer errors and which requires no more than \( 2st \) parity check digits.

---

Chapter 2. Coding Theory for Reliable Communications
Reed-Solomon codes are the special subclass of \( q^n \)-ary BCH codes with \( s = 1 \). The generator polynomial \( g(X) \) of \( t \)-error-correcting RS codes has roots that are \( d - 1 \) consecutive powers of a primitive element \( \alpha \), and is represented by

\[
g(x) = \prod_{i=1}^{2t} (x - \alpha^i).
\]  \hspace{1cm} (2.28)

These codes have the parameters shown in Table 2.2.

<table>
<thead>
<tr>
<th>Block length</th>
<th>( n = q - 1 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of parity check digits</td>
<td>( n - k = 2t )</td>
</tr>
<tr>
<td>Minimum distance</td>
<td>( d_{\text{min}} = 2t + 1 )</td>
</tr>
</tbody>
</table>

Theorem 2.16: An \( (n, k) \) RS code is a maximum-distance separable (MDS) code. A RS code has the very useful property that any \( k \) positions in a code word can be used as the information set. For any symbol positions, an \( (n, k) \) RS code on \( GF(q) \) has a code word corresponding to each of the \( q^k \) assignments in those \( k \) positions.

Theorem 2.17: The weight distribution \( \{W_i\} \) for an RS code over \( GF(q) \) can be obtained from the following formula found by Forney, Berlekamp, Peterson and Weldon (see Appendix C). For minimum distance \( d \),

\[
W_i = \binom{n}{i} (q - 1) \sum_{j=0}^{i-d} (-1)^j \binom{i-1}{j} q^{i-d-j} \quad \text{for} \quad d \leq i \leq n.
\]  \hspace{1cm} (2.29)

2.3.3.3 Binary Equivalent of Reed-Solomon Codes

Consider a \( t \)-error correcting \( (n, k) \) RS code over \( GF(2^m) \) with minimum distance \( d_{\text{min}} = n - k + 1 \). Since RS codes are linear cyclic codes, all possible code words, \( C \), can be generated by a matrix equation of the following type:

\[
C = u \cdot G.
\]  \hspace{1cm} (2.30)
where \( u \) is the \( k \) information symbol sequence \( \{ u_0, u_1, \ldots, u_{k-1} \} \) and \( G \) is the \( k \times n \) generator matrix with the cyclic form of the generator polynomial as shown in equation (2.20) in section 2.3.4.

Let \( \lambda_0, \lambda_1, \ldots, \lambda_{m-1} \) be the basis of \( GF(2^m) \) for binary representation (Definition 2.11). Any element \( \beta \in GF(2^m) \) can be expressed as

\[
\beta = \sum_{t=0}^{m-1} b_t \cdot \lambda_t \quad \text{for} \quad b_t \in GF(2) \tag{2.31}
\]

Then a code word vector \( c = (c_0, c_1, \ldots, c_{n-1}) \) is represented by

\[
c = \left[ (b_0^0, b_1^0, \ldots, b_{m-1}^0), (b_0^1, b_1^1, \ldots, b_{m-1}^1), \ldots, (b_0^{n-1}, b_1^{n-1}, \ldots, b_{m-1}^{n-1}) \right] \tag{2.32}
\]

where \( b_t^j \in GF(2) \). Thus the \( i \)-th symbol element of the code word, \( c_i \), can be written with the relationship of the generator matrix \( G \) and an information symbol sequence \( u = \{ u_0, u_1, \ldots, u_{k-1} \} \) as

\[
c_i = \sum_{j=0}^{k-1} u_j \cdot g_{ji} = \sum_{j=0}^{k-1} \left[ \sum_{t=0}^{m-1} b_t^j \cdot \lambda_{t+j} \right] \cdot g_{ji} = \sum_{j=0}^{k-1} \left[ \sum_{t=0}^{m-1} b_t^j \cdot \left( \lambda_t \cdot g_{ji} \right) \right] \tag{2.33}
\]

where \( g_{ji} \) is the symbol element in the \( j \)-th row and \( i \)-th column of the matrix \( G \). Since the term in the inner bracket in (2.33) becomes another linear combination, symbol elements of the matrix can be viewed at binary level as

\[
\begin{bmatrix}
\lambda_{m-1} \\
\vdots \\
\lambda_0
\end{bmatrix}
\cdot g_{ji} = 
\begin{bmatrix}
\lambda_{m-1} \cdot g_{ji} \\
\vdots \\
\lambda_0 \cdot g_{ji}
\end{bmatrix} \tag{2.34}
\]

Thus we have \( km \times nm \) binary generator matrix \( G^b \). With the binary generator matrix \( G_b \), each bit sequence of a code word, is represented by

\[
c_i^b = \sum_{j=0}^{km-1} u_j^b \cdot G_{ji}^b \quad \text{for} \quad i = 0, 1, \ldots, nm - 1. \tag{2.35}
\]
where \( u_j^b \) is the \( j \)-th bit of the information sequence and \( G^b_{ji} \) is the elements in the \( j \)-th row and \( i \)-th column of generator matrix \( G^b \).

If the generator matrix is in systematic form, the parity check matrix is easily obtained (see (2.15)) by Theorem 2.11. Thus we have the binary parity check matrix \( H^b \) as

\[
H^b = [h_0, h_1, \ldots, h_{m-1}]
\]

(2.36)

where \( h_i \), the \( i \)-th column vector of \( H^b \), has \((n - k) \cdot m\)-tuple.

### 2.3.3.4 Encoding of Binary Systematic Equivalents of Reed-Solomon codes

An encoding procedure to generate binary equivalents of \((n, k)\) RS codes with minimum distance \( d \), is as follows:

(i) Obtain the generator polynomial \( g(x) \) as in equation (2.28)

(ii) For the generator polynomial \( g(x) \), obtain the \( k \times n \) generator matrix \( G \) as in (2.20).

(iii) Convert the matrix \( G \) to systematic form \( G_s \).

(iv) Construct a binary matrix \( G^b \) by using equation (2.34)

(v) Generate code word \( C \) by the multiplication of information and generator matrices such that

\[
C = \begin{bmatrix}
  g_{0,0} & g_{0,1} & \cdots & g_{0,m-1} \\
  g_{1,0} & g_{1,1} & \cdots & g_{1,m-1} \\
  \vdots & \vdots & \ddots & \vdots \\
  g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,m-1}
\end{bmatrix}
\]

\[
\begin{bmatrix}
  c_0 \\
  c_1 \\
  \vdots \\
  c_{m-1}
\end{bmatrix}
\]

Once the systematic binary generator matrix is determined, the binary parity check matrix \( H_b \) is easily obtained by Theorem 2.11. Thus we have the binary parity check matrix \( H_b = [h_0, h_1, \ldots, h_{m-1}] \), where \( h_j \) is the \( j \)-th column of \( H_b \), which is an \((n - k) \cdot m\)-tuple.
An example of binary matrices for an RS code:
Consider an example for a two-error correcting (7,3) RS code over $GF(2^3)$ using a root of $p(x) = x^3 + x + 1$. The generator polynomials for these codes are

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

$$= x^4 + \alpha^2 x^3 + \alpha^5 x^2 + \alpha^5 x + \alpha^6.$$

By equation (2.20), the generator matrix $G$ is obtained as

$$G = \begin{bmatrix} 1 & \alpha^2 & \alpha^5 & \alpha^6 & 0 & 0 \\ 0 & 1 & \alpha^2 & \alpha^5 & \alpha^6 & 0 \\ 0 & 0 & 1 & \alpha^2 & \alpha^5 & \alpha^6 \end{bmatrix}$$

From this, we produce the systematic generator matrix $G_s$ as

$$G_s = \begin{bmatrix} 1 & 0 & 0 & \alpha & \alpha^3 & \alpha^6 & \alpha^6 \\ 0 & 1 & 0 & 1 & \alpha^4 & \alpha^5 & \alpha \\ 0 & 0 & 1 & \alpha^2 & \alpha^5 & \alpha^5 & \alpha^6 \end{bmatrix}$$

By (2.34), we get binary generator matrix $G_b$ and parity check matrix $H_b$ as

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

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

Chapter 2. Coding Theory for Reliable Communications
2.4. KEY CONCEPTS OF CODING SYSTEM

2.4.1 CHANNEL NOISE

The channel in the communication context consists of all the hardware and physical media from the modulator output, i.e. the transmitted signal $x(t)$, to the demodulator input, i.e. the received signal $y(t)$. As the transmitted signal passes through an additive noise channel, a random variable $n(t)$ is added as shown in Figure 2.1. Thus the received signal $y(t)$ is expressed by

$$y(t) = x(t) + n(t)$$  \hspace{1cm} (2.39)

![Figure 2.1 Additive Noise Channel](image)

When the distribution of the additive noise $n(t)$ is a Gaussian random variable with zero mean and one-sided power spectral density $N_o$, it is called an Additive White Gaussian Noise Channel (AWGN). The conditional probability of the output of $y$ given by an input of $x$, is

$$P(y|x) = \frac{1}{\sqrt{2\pi}\cdot\sigma} \exp\left(-\frac{(y-x)^2}{2\sigma^2}\right)$$  \hspace{1cm} (2.40)

where $\sigma^2$ is the variance of the noise input for each bit and is numerically equal to $\frac{N_o}{2}$. The AWGN channel model can be used as a typical model of satellite and deep-space links for which dominant noise is additive thermal or galactic noise.

In a fading channel, the amplitude of the transmitted signal varies randomly according to physical conditions. Rayleigh and Rician fading models are typical models of such channels.
The Rayleigh fading channel model arises from the combination at the receiver of many randomly phased scatter contributions. Each part of the scattered signal is a small fraction of the total received power.

As another fading model, the Rician fading channel model is considered in the case where there is one component which is stronger than the other scattered components. This means that the signals at the receiver consist of one component in a direct path and other scattered propagation paths.

These two modes have the probability distribution functions as follows (Rayleigh $f_{\text{Ray}}$ and Rician distribution $f_{\text{Ric}}$)

$$f_{\text{Ray}}(x) = \frac{x}{\sigma^2} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$  \hspace{1cm} (2.41)

$$f_{\text{Ric}}(x) = \frac{x}{\sigma^2} \cdot I_0\left(\frac{\mu x}{\sigma^2}\right) \exp\left(-\frac{(x^2+\sigma^2)}{2\sigma^2}\right)$$  \hspace{1cm} (2.42)

where $I_0(\alpha) = \int_0^{\infty} \exp(\alpha \cos \psi) \, d\psi$, $\alpha^2$ is the variance of either of the real or imaginary component of a complex Gaussian random variable representing a scattered signal and $\mu$ is the magnitude of a strong signal component.

### 2.4.2 DIGITAL CHANNEL MODEL

A digital channel model is conventionally a mathematical representation including the modulator and the demodulator. This channel model consists of a set of input symbols, output symbols, and transition probabilities. When it is assumed that the transition probabilities are time invariant and independent from symbol to symbol, it is called a memoryless channel. Furthermore the Discrete Memoryless Channel (DMC) is defined as a memoryless channel with $M$-ary input modulator and $Q$-ary output demodulator.

In particular, if the amplitude distribution of the channel noise is symmetric, a binary symmetric channel (BSC) is defined in the case where binary modulation, $M=2$, is used and demodulator output is quantized to $Q=2$. 

---

*Chapter 2. Coding Theory for Reliable Communications*
Let \( \{x_i\} \) be a set of \( M \)-ary input and \( \{y_j\} \) be a set of \( Q \)-ary output symbols. A DMC is clearly described by a set of transition probabilities \( P(y_j|x_i) \) which are conditional probabilities. A transition probability diagram is shown in Figure 2.2 and 2.3.

\[
\begin{array}{c}
0 \\
1 \\
\end{array}
\begin{array}{c}
\text{1-P} \\
\text{P} \\
\end{array}
\begin{array}{c}
\text{P} \\
\text{1-P} \\
\end{array}
\begin{array}{c}
0 \\
1 \\
\end{array}
\]

Figure 2.2 BSC

\[
\begin{array}{c}
0 \\
1 \\
\end{array}
\begin{array}{c}
P(0|0) \\
P(1|0) \\
P(0|1) \\
P(1|1) \\
P(Q-1|0) \\
P(Q-1|1) \\
\end{array}
\]

Figure 2.3 DMC channel with \( M=2, Q>2 \).

In the figures, the transition probability is derived from the specification of the transmitted wave forms, the signal power level, the transmission channel characteristics, and the description of the demodulator. Note that the DMC is not applicable for channels affected by an atmospheric impulse noise or inter-symbol interference.

2.4.3 ERROR BIT PROBABILITY IN AWGN

Let \( E \) be the transmitted signal energy per code word and \( E_s \) denote the signal energy for transmission of a single bit in the code word. If a code word consists of \( n \) bits with \( k \) bits of information, the energy per information bit is

\[
E_b = \frac{E}{k} = \frac{n}{k} E_s = \frac{E_s}{R}
\]

where \( R \) is code rate.

We consider a BPSK (Binary Phase Shift Keying) system in which the transmitted wave forms are represented by sinusoids of the same frequency with fixed phases 180° apart. Suppose that this system constitutes a BSC (Binary Symmetric Channel) with crossover probability \( P \) through the AWGN channel with zero mean and variance...
\[ \frac{1}{2} N_o. \] Let \( r_j \) represent the \( n \) sampled outputs of the matched filter for any particular code word given that \( t_j \) is a transmitted bit. Then the output \( r_j \) is expressed as
\[
\begin{align*}
    r_j &= \sqrt{E_s} + n_j \quad \text{for} \ t_j = 1 \\
    r_j &= -\sqrt{E_s} + n_j \quad \text{for} \ t_j = 0
\end{align*}
\] (2.44)
(2.45)

where \( n_j \) is additive white Gaussian noise. The error probability \( P \) on the AWGN is represented by
\[
P = Q\left( \frac{2E_s}{\sqrt{N_o}} \right)
\] (2.46)

where \( Q(\alpha) = \int_{\alpha}^{\infty} \frac{1}{\sqrt{2\pi}} \exp\left( -\frac{x^2}{2} \right) dx. \)

### 2.4.4 SOFT-DECISION DECODING

Suppose the system involves an \((n, k)\) block code employed with a BPSK system on a Gaussian channel. The demodulator provides \( n \) sequences which represent the actual sample voltage. Let \( Y \) be the received sequence and \( X_i \) be the transmitted sequence for \( 0 \leq i \leq 2^k - 1 \). An optimal decoder would choose the \( X_i \) which is the closest to the received sequence \( Y \). This is expressed by the conditional probability \( P(X_i|Y) \) that \( X_i \) is the transmitted sequence given that \( Y \) has been received. By Bayes’ rule, this probability can be expressed by
\[
P(X_i|Y) = P(Y|X_i) \frac{P(X_i)}{P(Y)}
\] (2.47)

If we assume that all messages are equally likely, then maximising \( P(X_i|Y) \) is equivalent to maximising \( P(Y|X_i) \). We can write \( P(Y|X_i) \) as the product of \( n \) Gaussian density functions as follows:
2.4 Key Concepts of Coding System

\[ P(Y|X_i) = \prod_{j=1}^{n} \left( \frac{1}{\pi N_0} \right) \exp \left( -\frac{(Y - X_j)^2}{2N_0} \right) \]  

(2.48)

Suppose that the \( M \)-ary input at the modulator is from the set \( X = \{x_0, x_1, \ldots, x_{M-1}\} \) and the possible demodulator outputs are \( Y = \{y_0, y_1, \ldots, y_{M-1}\} \). We consider input-output characteristics on the BSC. The conditional probabilities of (2.40) can be expressed as

\[ P(Y = y_j|X = x_i) = P(y_j|x_i) \]  

(2.49)

where \( i = 0,1 \) and \( j = 0,1,\ldots,q-1 \). Hence if a code word of \( u_0, u_1, \ldots, u_{n-1} \) is selected from \( X \), and the corresponding output is a sequence of \( v_0, v_1, \ldots, v_{n-1} \) from \( Y \), the joint conditional probability is

\[ P(Y_0 = v_0, \ldots, Y_{n-1} = v_{n-1} | X_0 = u_0, \ldots, X_{n-1} = u_{n-1}) = \prod_{j=0}^{n-1} P(y_j = v_j | x = u_j). \]  

(2.50)

A maximum likelihood decoder (ML decoder) is obtained by choosing the code word which maximizes the joint probability.

In practice, it is convenient to use the logarithm of the probabilities associated with the path of a code word. It is called the metric of the path. Thus the path metric \( M_p \) and bit metric \( M_b \) are expressed by

\[ M_b = \log \left( P(Y_j|X_i) \right) \]  

(2.51)

\[ M_p = \sum_{i=0}^{n-1} M_b^i \]  

(2.52)

2.4.5 CODING GAIN

A useful parameter is the ratio of energy per information symbol to noise spectral density to achieve a given probability of error. This term explains the amount of improvement that is achieved when a particular coding scheme is used. This is called coding gain. For example, in Figure 2.3, it is shown that the use of a particular coding scheme achieved 2.5 dB coding gain at bit error probability \( P_e = 10^{-6} \).
When we consider the merit of a particular code, *asymptotic coding gain* can be a measure, since it depends only on the code rate and minimum distance and can be defined for both an unquantized channel and a binary quantized channel (hard-decision).

**Figure 2.1 Coding Gain**

For a *BSC* with *AWGN* channel, for large \( \frac{E_b}{N_0} \), the asymptotic coding gain is well explained in [11][31]. As a result, the asymptotic coding gains are obtained as the following.

\[
G_a = 10 \log [R(t+1)] \quad \text{for hard-decision} \quad (2.53)
\]

\[
G_a = 10 \log [Rd] \quad \text{for unquantised soft-decision channel outputs} \quad (2.54)
\]

where \( R \) is code rate, \( d \) is the minimum distance, \( t \) is the number of errors which can be corrected, and \( t = \frac{d-1}{2} \). A comparison between equation (2.53) and (2.54) explains why the asymptotic coding gain with an unquantised demodulator is increased by around 3 dB over hard decision at high Eb/No. This is the significant advantage of soft-decision decoding over hard decision decoding, although the complexity increases because of more quantised inputs than in hard-decision.

*Chapter 2. Coding Theory for Reliable Communications*
2.5. CONCLUDING REMARKS

In this chapter, we have briefly discussed the mathematical background for coding theory, some specific linear cyclic codes and important terminology required in any communication system which uses coding. Further study can be carried out using one of many good books, some of which have been referred to here.
3.1 INTRODUCTION

Decoding methods for RS codes can be classified according to two types: algebraic and probabilistic decoding. Algebraic decoding utilizes the algebra of finite fields to search for the most likely pattern. Probabilistic decoding is the method which finds the code word with maximum probability compared with the received sequence. In probabilistic decoding, it is important to examine the state transitions between the input and output sequences of the encoder. A trellis is a convenient method of viewing the state transitions.

In this chapter, we review some notable soft-decision decoding approaches for RS codes with their trade-offs between complexity and performance. Then we present useful techniques for low complexity soft-decision decoding.

First of all, in section 3.2, there is a description of two soft-decision algorithms, GMD and the Chase algorithm, for which an algebraic decoding method is employed.

Section 3.3 is devoted to describing the fundamental properties of a trellis for block codes and a binary trellis structure for RS codes which is obtained from binary equivalents of RS codes.

In section 3.4, three trellis decoding approaches are discussed for RS codes. The first one is the Viterbi algorithm and the second is a reduced search algorithm which may be seen as a general form of the Viterbi algorithm. The third is a sequential decoding approach using a modified Fano algorithm. After this discussion we suggest some useful techniques required to develop low complexity sequential decoding.
In subsequent sections, a simulation test bed is explained and we show, by experiment, the variation of error occurrence according to different values of soft-decision. Finally the confidences for bit, symbol, and information block, are defined.

3.2. ALGEBRAIC DECODING FOR SOFT-DECISION

A big disadvantage of algebraic decoding algorithms has been a difficulty in employing channel measurement information, even though many researchers[22][36][53] have developed useful algorithms which can be used for RS codes with hard-decision decoding. However, Forney[18] and Chase[9] have individually presented soft-decision decoding methods for block codes. Here we discuss their basic methods.

3.2.1 FORNEY’S METHOD

This algorithm is also called the GMD (Generalized Minimum Distance) algorithm. In it, a distance measure is used allowing soft-decision information to be used in algebraic minimum distance decoding. The basic operation of this algorithm generates successive candidates for the decoder output until one of the candidates is chosen by a decoding criterion. The procedure of this algorithm is the following:

(i) Perform hard decision decoding on the received sequence.

(ii) By consideration of the reliability (soft-decision values), erase the two least reliable symbols and then do erasure-decoding.

(iii) Erase further two symbols and repeat until at most \( d_{\text{min}} - 1 \) symbols have been erased or the best code word has been found. The best code word is defined as the following way.

a) \( \Gamma_{\text{sum}} = 0 \).

b) If symbol \( i \) has confidence value \( \gamma \), then \( \Gamma_{\text{sum}} = \Gamma_{\text{sum}} + \gamma \) if the decoded and original symbols match. Otherwise \( \Gamma_{\text{sum}} = \Gamma_{\text{sum}} - \gamma \).

c) Obtained \( \Gamma_{\text{sum}} \) for entire code word, which consists of \( n \) symbols.

d) If \( \Gamma_{\text{sum}} \) is greater than \( n - d_{\text{min}} \), then the best code has been found.
If sufficient code words are generated, this algorithm can provide a very close approximation to ML decoding. However, it is difficult to find an efficient criterion which generates a set of code words including the correct code word. In fact, the performance of the GMD algorithm is poor compared with ML performance, since the criterion used is not optimal. Consequently, although the GMD algorithm has the advantage of low complexity decoding, the performance should be improved to achieve a desirable coding gain from soft-decision decoding.

3.2.2 CHASE ALGORITHM

Chase presented a soft-decision algorithm which can use channel measurement information. This algorithm generates a set of several candidate code words by using hard-decision decoding and chooses the most likely one among them. The basic procedure of the Chase algorithm is explained as the following:

(i) Make hard decision on each symbol in the received sequence to produce vector $Y$.

(ii) Deliberately generate various patterns of errors, $T$, to generate the test sequences $U=T+Y$. Decode each sequence created in this manner using hard decision decoder

(iii) Compute the distance from each of the code words to the received sequence, and then choose the code word with minimum distance.

This algorithm was classified into three methods according to the number of test sequences. The methods are described for binary codes; for the application to RS codes over $GF(2^m)$, the number of test patterns in $T$ should be increased because of non-binary symbols, since each symbol has $2^m$ possible combinations. The three methods are explained as follows.

**Method 1:** $T=\{\text{all error patterns with weight } \frac{d_{\text{min}}}{2}\}$.  

**Method 2:** $T = \{\text{all } 2^{(d_{\text{min}}/2)} \text{ combinations of values in the } \frac{d_{\text{min}}}{2} \text{ least reliable positions and zero in all other positions}\}$.  

---

Chapter 3. Soft-decision Decoding Methods for Reed-Solomon Codes
**Method 3:**

Determine the $d_{\text{min}} - 1$ least reliable symbols. The test patterns in $T$ are produced by putting one in the $i$ least reliable positions and zero in all other positions for $i = 0, 2, \ldots, d_{\text{min}} - 1$ for $d_{\text{min}} = \text{odd}$ and $i = 0, 1, 3, \ldots, d_{\text{min}} - 1$ for $d_{\text{min}} = \text{even}$. For the application of RS codes consisting of non-binary symbols, the number of least reliable symbols to be determined should be increased to

$$\sum_{i=0}^{d_{\text{min}}-1} (2^n)^i.$$ 

In Chase's results, algorithm 1 and algorithm 2 showed an approximate ML performance by using soft-decision information in addition to the use of a hard-decision decoder. Algorithm 3 is inferior in performance but uses a smaller number of test patterns. However, this algorithm still gives rise to a large number of computations for RS codes with large Galois fields due to a large number of test patterns.

### 3.3. TRELlIS STRUCTURE FOR BLOCK CODES

The first notion of a trellis diagram was introduced by Forney[20] to show that Viterbi's algorithm[57] was actually optimum. If the elements of an $N$-tuple code word $(c_0, c_1, \ldots, c_{N-1})$ are represented by a trellis diagram with $N$ sections corresponding to some path, then this trellis diagram is used as an efficient tool for the decoder to find, by orderly search, the code word with maximum path metric.

A trellis for block codes was presented for the first time by Bahl, Cocke, Jelinek, and Raviv[3] in 1974 as a method of representing the code words in an arbitrary linear block. In 1978, Wolf[59] showed that soft decision decoding could be implemented on a trellis for any $(n, k)$ block code with a maximum likelihood decoder.

Much theory about trellises is introduced in papers by Forney[16][17][21] and basic concepts are well explained in a tutorial paper by Muder[34] and in some books by Conway and Sloane[12][13]. There is also good descriptions in books published
recently by Honary and Markarian[58] and Lin et al[59] In this section, we study the fundamental concepts and present a new trellis scheme for RS codes.

3.3.1 TRELLIS FOR BLOCK CODES

The trellis of a block code can be defined as an edge-labeled directed graph \( T = (V, A, E) \) consisting of a set \( V \) of vertices, a set \( A \) of elements in a finite field and a set of \( E \) of edges. If \( T \) is a trellis of length \( n \) for a path corresponding to a code word \( c = (\alpha_0, \alpha_1, \ldots, \alpha_{n-1}) \), then \( T \) is represented by the graph

\[
\begin{align*}
  v_0 \xrightarrow{\alpha_0} v_1 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_{n-1}} v_{n-1}.
\end{align*}
\]

In binary block codes, each vertex has two edges directed from it, labeled 0 and 1, while non-binary codes over \( GF(q) \) have \( q \) different edges labeled from 0 to \( q-1 \).

Note that the edge and vertex are also called branch and node. In addition the vertex levels and edge levels are called depths and stages respectively.

Definition 3.1: Two trellises \( T \) and \( T' \) are isomorphic if there is a one-to-one correspondence \( f \) from \( V \) to \( V' \) such that \( f(V) = V' \), and \( e(v_i, v_{i+1}, \alpha) \) is an edge of trellis \( T \) if and only if \( e(f(v_i), f(v_{i+1}), \alpha) \) is an edge of trellis \( T' \).

Definition 3.2: A trellis \( T \) of length \( n \) is proper if \( V_0 \) has a single vertex \( O \) such that every vertex of the trellis lies on a length-\( n \) path from \( O \), and no two length-i paths from \( O \) corresponds to the same i-tuple. Thus, on a proper trellis, no two edges of the trellis \( T \) have the same initial vertex and the same label.

Definition 3.3: A proper trellis \( T \) for a code \( C \) is a minimal (proper) trellis if for every trellis \( T' \) for \( C \),

\[
|V_i| \leq |V'_i| \quad \text{for every } i
\]

where \( |V_i| \) is the number of vertices at level \( i \) in \( T \) and \( |V'_i| \) similarly for \( T' \).

Theorem 3.1: Every block code has a minimal proper trellis, and any two minimal proper trellises for the same code are isomorphic (proof:[40])
Definition 3.4: Let \( S = \max(|V_i|) \) where \( |V_i| \) is the number of states at level \( i \). The **minimal trellis size** is defined as an index \( s = \log_q(S) \).

### 3.3.2 TRELLIS COMPLEXITY

Trellis complexity of linear block codes was well discussed in\([12][13][17][27][28][29][40]\). Many measures of trellis decoding complexity have been introduced. We here consider three kinds of trellis complexity measures: **trellis state complexity**, **trellis branch complexity**, and **trellis edge complexity**. Let \( I \) be an ordered time axis or trellis level \( \{0,1,\ldots,n-1\} \) and \( q \) be the size of the code alphabet.

**Definition 3.5:** *Trellis state-complexity* \( S(C) \) is defined as

\[
S(C) = \max_{i \in I} \{ \log_q |S_i| \} \quad (3.3)
\]

where \( S_i \) is the set of states at level \( i \). Muder\([40]\) claims this complexity to be a fundamental descriptive characteristic of a given code.

**Definition 3.6:** *Branch-complexity* \( B(C) \) (by Forney\([17]\)) is defined as

\[
B(C) = \max_{i \in I} b_i \quad (3.4)
\]

where \( b_i \) is the logarithm of the number of branches in trellis at level \( i \). For Viterbi algorithm (see section 3.4.1), the total number of trellis branches per unit trellis level is usually regarded as a more accurate measure of decoding complexity than the number of states in the trellis\([29]\).

**Definition 3.7:** *Trellis edge complexity* \( E(C) \) (McEiece\([38]\)) is defined as

\[
E(C) = \sum_{i \in I} q^{b_i} \quad (3.5)
\]
McEliece concluded in his paper that this complexity, for a generalized version of the Viterbi algorithm, is proportional to the total number of branches or edges in the trellis.

### 3.3.3 WOLF’S TRELlIS FOR RS CODES

Wolf presented trellis structure which can be used for any \((n, k)\) linear block codes over \(GF(q)\). Thus we now discuss Wolf’s trellis for RS codes.

Consider \((n,k)\) Reed-Solomon codes over \(GF(2^n)\) having generator matrix \(G\) and parity check matrix \(H\). Let the \((n-k)\)-tuple \(h_i\) be the \(i\)th column of \(H\) for \(i = 1, 2, \ldots, n\).

Denote an arbitrary state at level \(i\) as \(v_i\):

\[
v_i = \sum_{j=1}^{l} b_j \cdot h_i
\]

where the path labeled by \((b_1, b_2, \ldots, b_n)\) corresponds to a code word. The trellis state sequence for an \((n, k)\) block code can be obtained by the following procedures.

(i) The initial state \(v_0\) has only one possibility \((0, 0, \ldots, 0)\): the all zero \((n-k)\)-tuple.

(ii) For a code word \(C = (c_0, c_1, \ldots, c_{n-1})\), \(H = (h_0, h_1, \ldots, h_{n-1})\), and \(i = 1, 2, \ldots, n\), trellis state \(v_i\) at level \(i\) is obtained by

\[
v_i = v_{i-1} + c_{i-1} \cdot h_{i-1}
\]

(iii) The trellis state sequence corresponding to a code word can be represented as

\[
v_0 \xrightarrow{\varepsilon_0} v_1 \xrightarrow{\varepsilon_1} v_2 \cdots \cdots \xrightarrow{\varepsilon_{n-1}} v_n
\]

Proposition 3.1: If a linear code is cyclic, the trellis state sequence is periodic. Thus an RS code has a periodic state sequence which includes the all zero state at depth 0 and at depth \(n\), that is \(v_0 = (0, 0, \ldots, 0)\) and \(v_n = (0, 0, \ldots, 0)\).

Proposition 3.2: If a linear code is cyclic, the reverse of any given state-sequence also exists as a state sequence.

Proposition 3.3: The minimal trellis size \(s(C)\) is upper bounded as
Proposition 3.4: For \((n, k)\) systematic codes, the state \(v_k\) at the end of the information sequence, indicates the unique remaining path which exists through parity check block.

We have explained in section 2.3.3.3 how to generate a binary equivalent of an RS code and have obtained the binary generator and parity check matrices. This binary parity check matrix is used to construct a binary trellis structure for the binary equivalent code of the RS code. In this trellis, the number of branches at each node is at most 2 and the trellis depth is longer than that of the trellis structure for the original RS code. This new trellis is called a binary-branch trellis in this thesis. Table 3.1 is a comparison of trellis structure between the binary branch trellis and the non-binary branch trellis for an \((n, k)\) RS code over \(GF(2^m)\).

<table>
<thead>
<tr>
<th>Possible Trellis States</th>
<th>Non-binary trellis</th>
<th>Binary trellis</th>
</tr>
</thead>
<tbody>
<tr>
<td>Possible paths</td>
<td>(2^mk)</td>
<td>(2^mk)</td>
</tr>
<tr>
<td>Trellis depth for information block</td>
<td>(k)</td>
<td>(mk)</td>
</tr>
<tr>
<td>Trellis depth for parity check block</td>
<td>(n-k)</td>
<td>(m(n-k))</td>
</tr>
<tr>
<td>Trellis length</td>
<td>(n)</td>
<td>(n \cdot m)</td>
</tr>
<tr>
<td>Number at branches</td>
<td>(2^n)</td>
<td>2</td>
</tr>
</tbody>
</table>

Figure 3.1 shows the binary-branch trellis for RS codes. As the figure shows, the first \(mk\) depths corresponds to information block and each node has binary branches. On the other hand, the last \(m(n-k)\) depths are associated with parity check block and each node has only a branch. Thus if the decoder reaches the end of information block, that is depth \(mk\), the remaining path can be predicted. This property can be useful in trellis search behaviour of a decoder, since the decoder does not need an alternative search during the parity check block.
Example:

We consider an example for (7,3) RS codes over $GF(2^3)$. Let an information sequence be (101001010). By the encoding procedure as described in section 2.3.3.4, the information sequence is encoded in binary and non-binary symbol form as

Symbol code word: $\alpha^6 1 \alpha \alpha^3 \alpha^5 \alpha^4 \alpha^2$

Binary code word: 101 001 010 011 111 110 100.

By the use of binary parity check matrix which has been obtained already in example 2.1 as
we can generate the trellis state at each depth according to the procedure we have explained in section 3.3.3. Let $b$ denote the state transition from a node to other node for a binary bit $b$. Each state is obtained from the systematic binary parity check matrix and thus the state transition of the trellis can be demonstrated as the following:

Information part: 000 000 000 000 $\rightarrow$ 011 111 010 010 $\rightarrow$ 011 111 010 010  
1 $\rightarrow$ 001 100 111 111 $\rightarrow$ 001 100 111 111  $\rightarrow$ 001 100 111 111 
1 $\rightarrow$ 000 010 011 101 $\rightarrow$ 000 010 011 101 $\rightarrow$ 011 111 110 100 
1 $\rightarrow$ 011 111 110 100 

Parity check part:  $\rightarrow$ 011 111 110 100 $\rightarrow$ 001 111 110 100 $\rightarrow$ 000 111 110 100 
1 $\rightarrow$ 000 011 110 100 $\rightarrow$ 000 001 110 100 $\rightarrow$ 000 001 110 100 
1 $\rightarrow$ 000 000 010 100 $\rightarrow$ 000 000 000 100 $\rightarrow$ 000 000 000 100 
1 $\rightarrow$ 000 000 000 000 $\rightarrow$ 000 000 000 000 $\rightarrow$ 000 000 000 000 

Figure 3.2 Trellis State Sequence for (7,3) RS codes

In Figure 3.2 it is shown that the state at the end of the information sequence is equal to the direction of the branches corresponding to the remaining path.

### 3.4. TRELlis Decoding Approaches for RS Codes

The use of a trellis seems attractive since the soft-decision values can be easily employed. Many decoding algorithms are introduced in [32]. In this section, trellis some search algorithms are discussed.
3.4 Trellis Decoding Approaches for RS codes

3.4.1 VITERBI ALGORITHM

Viterbi[57] showed that a maximum likelihood decoding procedure can be used practically for short-constraint-length convolutional codes. This algorithm chooses the path which is closest to the received path using trellis structure. Since this algorithm has been well described in many books, we briefly describe the algorithm’s procedure as the following.

(i) At a trellis level \(i\), obtain at each node the branch metric which is a measure to represent a difference between the possible branches and the received bit corresponding to the level \(i\).

(ii) As the trellis depth \(i\) is increased by one, calculate the path metric along all paths entering each node at the level. If there are more than one paths arrived at a node, then the path with the largest path metric is survived and stored the path.

(iii) If the decoder reaches a given level, stop. Otherwise, go to step (ii).

The Viterbi algorithm is optimum in providing maximum likelihood performance. However, this algorithm has serious limitations for very long codes over constraint length 10. These limitations are caused by the memory problem for storing encoder states and the heavy computations from all survival branches at each level.

3.4.2 REDUCED SEARCH ALGORITHM

In order to reduce such computational complexity, an attractive decoding method is a reduced search algorithm. The main idea of such an algorithm is that the decoder considers a section of the trellis containing the most probable paths.

Matis[37] presented a reduced complexity search algorithm for soft-decision decoding of linear block codes. In the algorithm he used a selected sub-trellis of the full trellis which has a much smaller width. That is, at certain given node levels, there are only the branches corresponding to hard-decision receiver outputs which are the most reliable. Let the number of reliable information positions be given by \(k-p\) for \((n, k)\) block code over \(GF(q)\) where \(p\) is positive integer. If the correct path is chosen as the most likely one among a set of code words which all agree with the hard-decision.
Otherwise all combinations of digits in the \( p \) remaining positions are considered. In this algorithm, the decoding complexity depends on the value of \( p \). If \( p \) is zero, it accepts simply hard-decision outputs. The choice of \( p=k \) corresponds to ML decoding, since all possible paths, \( q^{n-k} \), are tried. In this work, the trade-off between computational complexity and bit error performance was demonstrated.

Another approach to a reduced search algorithm was introduced by Shin[52]. This is to reduce the most likely paths at each level. According to the method of reducing paths, three methods were introduced: Search Algorithm with \( B \) active states, Reduced search by path metric, and Reduced search by branch metric. The objective of the above methods is to improve the computational complexity by reducing the searching region to smaller portion including a most likely path. The methods are briefly introduced.

**Method 1: Reduced Search with \( B \) Active States**

This method is to limit the states at each level by choosing a fixed number \( B \) of paths at a level which then are passed to next level for \( 1 < B < S \), where \( S \) is the number of all possible states. In order to select the best \( B \) paths at every level, a sorting process is carried out with respect to path metric. If the surviving paths with the same path metric are more than \( B \), arbitrary \( B \) paths are taken. This method can be said to be a modified Viterbi decoding with reduced paths and sorting process. Although this method can reduce the complexity of Viterbi decoding, in very large \( B \) the sorting program can be more complicated and less efficient than the full search method.

**Method 2: Reduced search by path metrics**

The inefficiency from the sorting process in method 1 can be improved by the reduced search by path metrics (RSP). A threshold value (or reference metric) is determined according to the \( SNR \) (signal-to-noise-ratio) of a given channel. All survival paths are reduced by comparison with the reference path metric at each level. By the reference path metric, the correct path can be included in the survival paths. In order to approach the performance of Viterbi decoding, the reference path metric is adjusted by decreasing or increasing the value.
Method 3: Reduced Search by Branch Metric
This method uses branch metric to reduce the selected paths at a level. The confidence region is decided according to quantised level. According to the region, confidence level (CFL) is assigned to each region and the maximum distance with respect to a given confidence level is determined. Maximum branch metric is obtained for each symbol of the received sequence. This branch metric is used to reduce the search paths. Suppose that a confidence level is set. If there are no paths less than the maximum branch metric, the confidence region is reset. When the survival paths are too large at low confidence region, the paths are reduced to a fixed number in order of path metric.

Reduced-search soft-decision trellis decoding can be regarded as a generalization of the Viterbi algorithm. It can reduce the complexity required for the correct path search. However, this approach has not been able to achieve sufficient performance compared to ML decoding. There are trade-offs between computational complexity and bit error performance.

3.4.3 SEQUENTIAL DECODING FOR RS CODES
Sequential decoding was firstly introduced by Wozencraft for convolutional codes in 1957[60]. A few years later, Fano[14] presented a refined version of the sequential decoding which is called the Fano algorithm in 1963 and then Zigangirov[61] and Jelinek[26] discovered separately another algorithm called the Stack Algorithm.

This decoding method is a sub-optimum method which explores a small subset of all possible paths to search for the correct path in a sequential manner. If the explored path at a given stage looks promising, it is explored further; otherwise the path is rejected and another path is tried.

Shin[52] introduced a sequential decoding method for RS codes using a modified Fano algorithm and showed useful coding gain over hard decision. In this method the encoder is as shown in Figure 3.3 and the trellis is constructed by tracing possible states of the encoder memory for all possible inputs. On the trellis there are $2^n$ incoming and outgoing branches at each node for $(n, k)$ RS codes over $GF(2^n)$.
Shin used a modified Fano algorithm to implement the application to RS codes. The basic operation of the Fano algorithm is to move forward or move backward from a node to the next node according to the comparison of a postulated path with a threshold. If the current path satisfies the threshold, the decoder moves forward from a node to next node. Otherwise the decoder moves backward to search for alternative branches which have not been tried. This operation is briefly called the *backtracking operation* in the thesis.

The modification by Shin's approach includes two other important thresholds, namely $T_n$ and $T_a$. The value of $T_n$ is chosen empirically to decide an acceptance of a path at the end of the trellis. The value of $T_a$ is empirically obtained to reject unlikely alternative paths. In this algorithm, the choice of $T_n$ and $T_a$ are very critical factors which influence the decoding performance and complexity. She demonstrated considerable coding gain at reasonable complexity compared to previous work. However, her sequential decoding approach has several drawbacks.

Firstly, this approach based on symbol level cannot fully employ bit-level soft information, since the branch transition at each node should be carried out with symbol level soft-decision information which consists of $m$ individual bits. In a probabilistic decoding, the loss of bit-level soft-decision information can influence the efficiency of searching operation. Thus it is desirable to use a trellis structure with branch transitions at the bit level. However the approach cannot be extended to equivalent binary codes because they are not in general cyclic codes.
Secondly, in consideration of the basic operation of the Fano algorithm, the non-binary branches can sacrifice the strength of the algorithm with powerful backtracking behaviour. The reason is that the decoder should do a comparison process at each node in the backtracking operation to choose the best of $m$ non-binary branches. In this point, it may be more complex to determine the most proper value of $T_n$ in the application to RS codes.

In summary, the idea of this approach by Shin is reasonable in that it uses the trellis structure for soft-decision decoding and the Fano algorithm was well chosen as an alternative to the Viterbi algorithm. However, the three drawbacks mainly caused by non-binary branches should be solved to improve decoding performance and reduce complexity. In the next section, we suggest a new approach to overcome those drawbacks.

### 3.5. LOW COMPLEXITY BIT-LEVEL SOFT-DECISION DECODING FOR RS CODES

We have examined tradeoffs of previous approaches for soft-decision decoding of RS codes. In this section, we present some schemes for a low complexity trellis decoding. Our approach intends to achieve a soft-decision decoding method for RS codes satisfying the following:

1. To provide easy implementation
2. To make full use of bit-level soft-decision information.
3. To have the lowest complexity without loss of decoding performance

For such a decoder, we consider three essential techniques in this thesis as follows.

- Construction of binary-branch trellis by Wolf’s method
- Sequential decoding.
- Permutation Decoding

These schemes will be explained in more details in later chapters of this thesis. We here describe the fundamental concepts.
3.5 Low Complexity Bit-level Soft-decision Decoding for RS codes

3.5.1 BINARY-BRANCH TRELLIS STRUCTURE

Wolf’s method for the construction of block codes is used for RS codes. Since this trellis is constructed by the use of systematic parity check matrix for RS codes, it is simple to implement. Furthermore, in order to employ bit-level soft-decision information fully, the binary parity check matrix can be obtained as in section 2.2.3.4. In this structure, the trellis state at each node can be obtained by referencing the column vector of the binary parity check matrix corresponding to each bit of the received sequence. As the decoder moves forward on the trellis, the next state is easily generated. Thus the sequential decoder stores the current state and the previous states and these two states are updated as the decoder moves forward or backward. This method can save considerable memory for the application to RS codes with large Galois fields and low code rate. Furthermore, since the use of binary branches gives only one alternative branch, back tracking operation is simplified without any complex procedure to qualify an alternative path in a non-binary branch, as shown in Shin’s method[52].

3.5.2 MODIFICATION OF FANO ALGORITHM

As a trellis search algorithm, we choose the Fano algorithm, modified for the improvement of error correcting performance.

The first modification contains a decision rule to decide whether to accept a path searched at the end of a trellis, similar to the threshold $T_n$ in Shin’s method. This decision rule reduces the decoding errors by a wrong path search when the decoder has chosen incorrectly. As the decision rule, a lower bound for a correct path proposed by Fano[14] is used.

Another modification is the path updating function to achieve maximum likelihood performance with respect to paths that the decoder has tried at least once. The role of this function updates the best path whenever a path has been searched. When there has been no valid path satisfying a given decision rule, the decoder releases the last updated path which is the most likely path amongst the paths which have been tried so far. This function is effective to protect the decoder from missing the correct path in
severe noisy channel conditions even though the path has been searched already. Compared to the modification by Shin, this new approach no longer needs the threshold $T_n$ to pick up the best branch among non-binary branches.

### 3.5.3 PERMUTATION DECODING

We have suggested a bit-level soft-decision decoding using a modified Fano algorithm in the above sections. However, the decoding complexity of this method would be reduced further for the application of long RS codes with a large number of trellis states although it provides considerably low complexity for some short RS codes.

Sequential decoding performance will depend on how efficiently the decoder searches for the most likely region of the trellis. However, the decoder may digress for such a region because of an undesirable combination of error bits. Furthermore, a long burst of noise is expected to require a large number of computations. The reason is that the decoder will do many more computations during the section corresponding to the burst error until the decoder can extend a further path over that section of trellis.

In consideration of this situation, it will be effective to change the bit order of the received sequence. Many schemes for this are suggested in [4][27][40][48]. In this thesis, we present three permutation techniques, cyclic, squaring, and hybrid permutation. Since these permutation groups are obtained from some mathematically useful properties of RS codes, a permuted sequence is seen as decoding to another code word. Therefore, the sequence can be directly applied to the modified Fano algorithm. These permutation decoding methods will be explained in chapter 5 and chapter 6.

### 3.6. SIMULATION TEST BED

A simulation test bed is used to evaluate the performance of a particular decoding scheme. In this thesis, the simulation program is written in the programming language C. We describe the simulation structure and a modeling method for a communication system which is used in this thesis. As Figure 3.4 shows, the simulator structure has four main modules as follows;
3.6 Simulation Test Bed

3.6.1 USER INTERFACE
In User Interface, the parameters for a particular simulation are fixed. The parameters include code length and information length, a range of Eb/No (*signal-to-noise ratios*), the total number of tested code words, and number of quantization levels from a demodulator.

3.6.2 INITIALIZING MODE
In this module, initial data which is needed in the simulation are generated. If an RS codes is decided in User Interface module, the generator polynomial and generator matrix are generated. Then this matrix is changed to systematic form by an algorithm which is described in Appendix D. Then the parity check matrix is easily obtained as shown in section 2.3.3.3.
3.6.3 DECODING SIMULATION

We describe the implementation of the communication model for simulation. In a BPSK system, two signal waveforms $s_1(t)$ and $s_2(t)$ corresponding to binary data, 0 or 1, are generated in the interval $0 \leq t \leq T_b$. The continuous waveforms are modeled to discrete from with 16 samples during a given $T_b$ as shown in Figure 3.5.

![Figure 3.5 Signal Representation for BPSK system](image)

It is assumed that the two modulated signals are equally likely to be transmitted into the channel having Gaussian noise with zero mean and variance $\sigma^2 = \frac{N_0}{2}$. Thus each sampled element has distortion due to the additive noise and the waveform for each symbol which is viewed at the demodulator by $r_i = s_i + n_i$ for $1 \leq i \leq 16$.

The demodulator detects the received signal using a squaring detecting method and then the detected signal is quantized to $Q$-levels according to a fixed threshold boundary. In the simulator, an 8-level uniform quantisation scheme suggested in Clark and Cain[11] is employed and it is refined for 16-level quantisation as shown in Figure 3.6 and Figure 3.7, such that $\Lambda = \sqrt{\frac{E_p}{N_0}} R$ where $R$ is code rate.

![Figure 3.6 8-Level Quantization](image)

![Figure 3.7 16-Level Quantization](image)
3.6.4 PERFORMANCE EVALUATION

This module includes the sub-modules representing encoder, modulator, Gaussian channel model, demodulator with quantizer, and decoder. The simulation is repeated for the number of code words to be tested.

The performance of a decoding scheme is measured as a function of $Eb/No$ in terms of error correcting performance and complexity. The error correcting performance is estimated by BER(bit error rate), $P_e$, obtained as

$$BER(P_e) = \frac{\text{Incorrectly decoded bits}}{\text{Total tested bits}}.$$  

The number of total tested bits is chosen to be at least 100 times the desired value of BER, so as to achieve a statistically reliable result. Since the simulator calculates the current BER at every transmitted sequence, it is possible to monitor the variations of the value of BER and to terminate the simulation when these fall below 5%.

A typical computation in sequential decoding involves extending code branches, finding the branch metrics, computing the path metric, and choosing the best path metric. In this thesis, one computation unit is defined as one path extension. Thus it seems reasonable that the complexity is measured by average path extension per information bit, $C$. In permutation decoding, there is additional complexity cost, which is to assign the priority of possible sequences. However, overall complexity is dominated by the algorithm complexity of the sequential decoding since the extra cost requires simply a small number of sequence shifts and sorting operations.

3.7 BIT RELIABILITY BY SOFT-DECISION

In the communication system, the demodulator has the role of estimating a transmitted symbol based on an observation of the received signal. In a coded system, the output of the demodulator is quantized to give the decoder the reliability for the decision on each bit. Thus this soft-decision value can be an important clue for a decoder which considers the bit reliability. We can verify this statement by a simple experiment.
For the experiment, when a code word is transmitted through Gaussian channel, the demodulator detects a received signal and quantizes the received signal to 8 and 16 levels. In order to obtain the probability by wrong decision at each quantized level, 10,000 code words are tested.

![Error Occurrence by Quantization (8-level soft-decision values)](image1)

Figure 3. 8 Bit Error Rate for 8-Level Quantization

![Error Occurrence by Quantization (16-level soft-decision values)](image2)

Figure 3. 9 Bit Error Rate for 16-Level Quantization
Figure 3.8 and Figure 3.9 show the result with 8 and 16-level quantization. As the figures show, the strong signal has very low bit error probability. Thus the soft-decision information can be seen as a statistical measure indicating the confidence that a bit has been received correctly. Since the soft-decision decoding provides more channel measurement information than hard-decision decoding, the confidence of each bit can be reflected in the decoding process so that the decoder can use more reliable bits. This validity is clearly shown by the experiment.

Definition 3.8: We define bit confidence as the positive number corresponding to quantized modulator output (soft-decision) as shown in Table 3.1 and 3.2:

<table>
<thead>
<tr>
<th>Quantized level</th>
<th>7</th>
<th>0</th>
<th>6</th>
<th>1</th>
<th>5</th>
<th>2</th>
<th>4</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit confidence</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Quantized level</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>2</th>
<th>1</th>
<th>3</th>
<th>1</th>
<th>4</th>
<th>1</th>
<th>5</th>
<th>1</th>
<th>6</th>
<th>9</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit confidence</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Definition 3.9: Symbol confidence \( f \) is defined as

\[
f = \min B_i \text{ for } 0 \leq i \leq m - 1
\]

where \( B_i \) is the confidence of the \( i \)-th bit consisting of each symbol of RS code.

Definition 3.10: For \((n, k)\) RS codes, Information Block Confidence (IBC) is defined as the sum of confidences of \( k \) information symbols such as

\[
I = \sum_{j=0}^{k-1} f_j
\]

where \( f_j \) is the symbol confidence level of the \( j \)-th symbol of the received sequence.
3.8. DISCUSSION

In this chapter, we have reviewed the decoding methods for RS codes and have discussed some techniques required to achieve low complexity soft-decision decoding. In consideration of decoding complexity and decoding performance, a trellis decoding approach gives an easy implementation for soft-decision decoding and sequential decoding is a desirable alternative to Viterbi decoding. Moreover permutation techniques will contribute to further complexity reduction. In the following chapters, more details will be explained.
CHAPTER 4

SEQUENTIAL DECODING APPROACH

FOR REED-SOLOMON CODES

4.1. INTRODUCTION

In this chapter, sequential decoding is further studied and a new sequential decoding method is presented for application to RS codes.

Firstly, a number of sequential decoding algorithms are surveyed in section 4.2 and some requirements for a desirable sequential algorithm are discussed and these are considered as a strategy for choosing an efficient sequential decoding algorithm.

Secondly, there is a description of metric as a measure of likelihood for sequential decoding, which is essentially used for comparing paths. The complexity problem of sequential decoding is discussed. Next in section 4.5, we describe two typical sequential decoding algorithms, Stack and Fano algorithm with analysis of their tradeoffs for practical applications. Then error events of sequential decoding are discussed. These are important to analyse the performance of sequential decoding compared with ML performance.

Thirdly, we present a modified Fano algorithm in which the standard Fano algorithm is modified to improve the decoding performance. This algorithm is implemented on a binary-branch trellis based on Wolf’s method. On the binary-branch trellis, the metric for each bit is applied so that bit-level soft-decision can be fully employed.

Finally, some simulation results are shown in terms of error probability and complexity. In these results, several sensitivities which affect the behaviour of the Fano algorithm, are verified.
4.2. A SURVEY OF SEQUENTIAL DECODING ALGORITHM

Anderson and Mohan[1] analysed the cost of many sequential algorithms in terms of various measures including the number of branches, the size of storage, and the access cycle to the storage. They classified trellis searching schemes as sorting or non-sorting, and as breadth-first, metric-first, and depth-first according to the search method of a code tree (trellis). These schemes are summarized in Table 4.1.

Table 4.1 Search Algorithm Classification[1]

<table>
<thead>
<tr>
<th>Metric-first</th>
<th>Breadth-first</th>
<th>Depth-first</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sorting Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Stack Algorithm</td>
<td>M-Algorithm</td>
<td></td>
</tr>
<tr>
<td>Merge Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bucket Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Haccon's Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Muti-Stack Algorithm</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Non-Sorting Algorithm</td>
<td>Simmon-Wittke</td>
<td>Single Stack Alg.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Fano Algorithm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2-Cycle Algorithm</td>
</tr>
</tbody>
</table>

The depth-first algorithm has a search method that a decoder explores along the depth of the tree pursuing a single path until the metric of the path falls below a threshold. When a path fails to satisfy the threshold, the decoder moves backward to the first unexplored path and continues to search for the correct path from the path. This is called a backtracking operation. The Fano-algorithm is a well-known depth-first algorithm without sorting.

In breadth-first algorithms, all branches are searched at a given depth before moving on to the next depth and there are no backtracking searches. Viterbi[57] and the M-algorithm [30] are included in this kind of algorithm.
The **metric-first algorithms** have a search method that the next path extended is always the one with the best metric among the paths currently stored. Since this algorithm usually has a sorting process to single out the best path, there is at any time a best path at each stage. The Stack algorithm[26][61] is a typical metric-first algorithm.

In the cost analysis by Anderson and Mohan[1], some conclusions were made to decide on the best sequential algorithm. The use of sorting requires more cost than that a non-sorting and metric-first search is less efficient than bread-first or the depth-first. In a later section, we shall describe two well known sequential algorithms: The Stack algorithm and The Fano algorithm.

### 4.3. DEFINITION OF METRIC IN SEQUENTIAL DECODING

We have generally defined some metrics in section 2.4.4 which can be applied to Viterbi algorithm. In sequential decoding, a metric is also used as a basic means to compare paths. However, we need further consideration for application to sequential decoding. The reason is that while the length of paths, which are compared at any trellis level, are the same in the Viterbi algorithm, the sequential decoder has to compare paths with many different lengths. Thus these metrics will need to be adjusted for sequential decoding.

Fano[14] originally proposed a metric, the **Fano metric**, for sequential decoding and later Massey[35] proved that the Fano metric is the best choice. Therefore we now introduce the Fano metric which will be used in all the work of this thesis.

For \( (n, k) \) block codes, suppose that a code word \( y_i = \{y_0, y_1, \cdots, y_{n-1}\} \) is transmitted and a received sequence \( r_i = \{r_0, r_1, \cdots, r_{n-1}\} \) is obtained. The **bit metric** \( M_b(i) \) for the \( i \)-th bit is defined as

\[
M_b(i) = \log_2 \frac{P(r_i|y_i)}{P(r_i)} - R \tag{4.1}
\]

where \( R \) is code rate, \( P(r_i|y_i) \) is the probability of observing the \( i \)-th bit in the received sequence given that \( y_i \) is transmitted and \( P(r_i) \) is the total probability of observing an output in the received sequence.
4.3 Definition of Metric in Sequential Decoding

If a trellis has an associated branch of \( l \) bits, a branch metric \( M_b \) at a node of level \( k \) is

\[
M_b(k) = \sum_{j=0}^{l-1} M_b(j + l \cdot (k - 1)), \tag{4.2}
\]

and the path metric \( M_p \) is simply

\[
M_p = \sum_{i=1}^{n} M_r(i) = \sum_{i=1}^{n} M_b(i). \tag{4.3}
\]

Assume that the code word \( y \) is transmitted with equal error probability through the AWGN channel with zero mean and variance \( \sigma^2 = \frac{N_0}{2} \). For a BPSK system with binary input 0 or 1 and a received sequence \( r_i = \{r_0, r_1, \ldots, r_{n-1}\} \) with \( Q \)-ary output \( j = 0, 1, \ldots, Q - 1 \) between boundary \( I_j \) and \( I_{j+1} \), the probability \( P(j | b = 0, 1) \) observing the bit \( b \) in the \( j \)-th quantization level is represented by

\[
P(j | b = 0, 1) = \frac{1}{\sqrt{2}} \int_{t_j}^{t_j'} \exp\left(-\frac{1}{2}a^2 x^2\right) dx \tag{4.4}
\]

where \( a = \frac{2 \cdot E_s}{N_0} = \sqrt{\frac{2 \cdot R \cdot E_b}{N_0}} \). The total probability observing an output in the \( j \)-th quantization is

\[
P(j) = \frac{1}{2} \left[ P(j | 0) + P(j | 1) \right] \tag{4.5}
\]

Therefore the bit metric \( M_b(i) \) in equation 4.1 with respect to input \( b_i = 0, 1 \) at \( j \)-th quantization level, is

\[
M_b(j) = \log_2 \left( \frac{P(j | 0, 1)}{P(j)} \right) - R \tag{4.6}
\]

Furthermore we define positive bit metric \( + M_b(j) \) and negative bit metric \( - M_b(j) \) as

\[
+ M_b(j) = \log_2 \left( \frac{P(j | b)}{P(j)} \right) - R \quad \text{for} \; y_i = r_i \tag{4.7}
\]

\[
- M_b(j) = \log_2 \left( \frac{P(j | \overline{b})}{P(j)} \right) - R \quad \text{for} \; y_i \neq r_i \tag{4.8}
\]

where \( \overline{b} \) is the binary complement of input \( b_i \in \{0, 1\} \). Table 4.1 shows the probabilities and metrics for (7,3) Reed-Solomon codes with \( Q=8 \) at \( \frac{E_b}{N_0} = 4.0 \) dB.

---

Chapter 4. Sequential Decoding Approach for Reed-Solomon Codes
4.3 Definition of Metric in Sequential Decoding

| j  | \( P(j|0) \)         | \( P(j|1) \)         | \( P(j) \)         | \( + M(j) \) | \( - M(j) \) |
|----|-----------------|-----------------|-----------------|-------------|-------------|
| 0.7| 0.750387        | 0.000709        | 0.375548        | 0.57        | -9.48       |
| 1.6| 0.112843        | 0.002078        | 0.057460        | 0.55        | -5.22       |
| 2.5| 0.071780        | 0.006533        | 0.039157        | 0.45        | -3.01       |
| 3.4| 0.038398        | 0.017272        | 0.027835        | 0.04        | -1.12       |
| Sum| 1.000000        | 1.000000        | 1.000000        |             |             |

4.4. COMPLEXITY PROBLEM OF SEQUENTIAL DECODING

The problem with sequential decoding is that the number of computations required to extend a node is a random variable. For such a computational problem, Jacobs and Berlekamp [25] found an important result that the distribution of computation is lower bounded by Pareto distribution:

\[
P[C > L] = \Lambda \cdot L^{-\alpha}
\]

where \( C \) is the number of branches which the sequential decoder investigates to find the correct path, \( \Lambda \) is the number of the transmitted information bits, \( \alpha \) is a number depending on the channel and code rate \( R \). This result is applicable to any sequential decoding with the following two properties:

(i) the branches are examined sequentially so that at any node of the tree the decoder's choice among a set of previously unexplored branches does not depend on received branches deeper in the tree.

(ii) the decoder performs at least one computation for each node of every examined path.

In order to derive equation (4.9), assume that there is a discrete memoryless channel with input and output symbols. Suppose that the first \( N \) symbols are received as erasures which indicate the reception of signals whose symbol values are in doubt. Let each one have the non-zero probability \( W \) whatever input symbol is transmitted. The probability of erasure burst is \( W^N \) and the number of all possible paths is \( 2^{NR} \). When the sequential decoder
selects a path among those paths, there is probability $\frac{1}{2}$ that it must examine at least $\frac{1}{2}$ of them before locating the correct path. Since those examinations require at least one computation per path, total computations $L$ are at least $\frac{1}{2} \cdot 2^{N/2}$. Here the probability $P[C > L]$ is given by

$$P[C > L] = \frac{1}{2} W^N = \frac{1}{2} \cdot 2^{(N/2) \log_2 W} = \frac{1}{2} (2L)^{(\log_2 W)R}$$

$$= a L^{-\alpha} = \left( \frac{\alpha}{\Lambda} \right) \cdot \Lambda L^{-\alpha}$$

(4.10)

where $\alpha = -\log_2 (W/R)$ and $a = 2^{(1+\alpha)}$. This distribution of the form $a L^{-\alpha}$ is called Pareto with exponent $\alpha$.

By the above result, we can observe two interesting points in the computational behavior of sequential decoding. One is that the computations required to find the correct path through a burst error increases exponentially with the length of the burst $N$. The other is that the probability of a noise burst $N$ decreases exponentially with $N$. The combined effect of those effects produces the Pareto distribution of the computation. In addition, the property implies that sequential decoding is very sensitive to a burst error channel, in which the probability of a burst error of length $N$ does not decrease exponentially with $N$.

Consequently these results explain that the channel condition and the code rate $R$ are the parameters which influence the computations of the sequential decoding. Moreover as the values of $L$ is relatively large, the distribution is dominated by the term of $L^{-\alpha}$ so that the equation (4.9) can be approximately obtained from equation (4.10).

Sequential decoding can be easily applied to block codes as a special case. Let $\{y_i\}$ be the set of binary block codes with length $N$, information length $K$ and code rate $R$. Let $r$ denote a received sequence and $M$ be all possible code words such that $M = 2^{KN}$. Arrange the $M$ probabilities $P(r|y_i)$ in order of decreasing size. Let $I_b(r)$ denote the set of indexes of the last $M - L$ least likely sequence. The joint probability of transmitting a $y_i$ with an index in $I_b(r)$ and receiving $r$ is given by

$$\sum_{i \in I_b} P(r|y_i)$$

(4.11)
4.4 Complexity Problem of Sequential Decoding

The error probability for a maximum likelihood list of length $L$ can be written by

$$P(L_e) = \sum_{\text{all } r} \left( \sum_{i \in \mathcal{L}} \frac{1}{M} P[r|y_i] \right)$$

(4.12)

4.5. SEQUENTIAL DECODING FOR BLOCK CODES

The sequential decoding algorithm can be used as a trellis search algorithm for block codes. However, a little different interpretation compared with convolutional codes is required because of the inherent characteristic of the trellis. Since $(n, k)$ block codes over $GF(q)$ have $M$ code words with $n$ length, where $M = q^k$, the constraint length used in the convolutional codes, is replaced with code length $n$ for block codes. Furthermore, the sequential decoder does not need an alternative search during the last $(n-k)$ stages in the trellis based on Wolf’s method. The reason is that, as we have mentioned in chapter 3, each path arriving at the end of information block, depth $k$, has one unique path reaching the last single state. In the following section we discuss the two best known sequential decoding algorithms: Stack and Fano algorithm.

4.5.1. Stack Algorithm

Stack algorithm keeps candidates paths on a memory stack, resorting the stack according to path likelihood (path metric), and proceeds along the currently best path. Thus this algorithm is a kind of metric first algorithm with sorting processor. The basic operations of the stack algorithm are as follows:

(i) Load the stack with the origin node in the tree, whose metric is taken to be zero
(ii) Compute the metric of the successors of the top in the stack.
(iii) Delete the top from the stack.
(iv) Insert a new successor in the stack, and rearrange the stack in order of decreasing metric values.
(v) If the top path in the stack ends at a terminal node in the tree, stop.
(vi) Otherwise, return to step (ii).

The stack algorithm has a simple structure and there are some interesting characteristics. One of them is that any node is never visited more than once. Another thing is that the decoder can jump between nodes, although it cannot move to the previous node of the
current path. This flexibility can reduce the number of computations to find out the correct path. However, the stack algorithm contains two undesirable features. The first one is the process of reordering the stack at every node. The second one is that the size of stack increases as the decoder advances along the correct path so that the decoder must keep the paths which have been visited so far at every stage. If the information length is large, the increment of stack size and the reordering will give rise to a serious computational problem.

![Flowchart of Stack Algorithm](image)

**Figure 4.1 Stack Algorithm**

### 4.5.2. Fano Algorithm

Fano algorithm exploits the metric function which has the property that the metrics of the correct path tends to increase at a rate, while the metrics along incorrect paths decrease relatively quickly. In this algorithm, the correct path is found by comparing the metric along a postulated path with the current threshold \( T \) that may be loosened or tightened by adjusting with a uniform spacing \( \Delta T \). For such a proper operation of this algorithm, the Fano metric is used as shown in equation (4.6).
We define the decoding variables and constants of the Fano algorithm for block codes, as shown in Table 4.2.

Table 4.2 Variables and Constants of Fano Sequential Decoding

<table>
<thead>
<tr>
<th>Variables</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Lambda_l$</td>
<td>Path metric at trellis depth $l$</td>
</tr>
<tr>
<td>$T$</td>
<td>Current threshold</td>
</tr>
<tr>
<td>$\lambda_{(l+1)}(l+1)$</td>
<td>The metric of the $(l+1)$-th extension to the node at the depth $l+1$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Constants</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Delta T$</td>
<td>Threshold spacing value</td>
</tr>
<tr>
<td>$\Lambda_0$</td>
<td>Initial threshold value</td>
</tr>
<tr>
<td>$L$</td>
<td>Overall computational limit</td>
</tr>
</tbody>
</table>

We consider Fano algorithm for $(n, k)$ block codes. As Figure 4.2 shows, the basic operations of the Fano sequential decoder are as follows:

(i) The initial threshold $T, \Lambda_0$, and current depth $l$ are initialized to zero.
(ii) Set flags $F_r(l)$ and $F_a(l)$ to zero, which mean that current mode in non-threshold violation mode and the current node has an alternative path.
(iii) Compute path extension from a node of the current depth $l$, which is the process to calculate the path metric $\Lambda(l+1)$ up to the extended path. If the current node is visited at the first time, that is $F_a(l) = 0$, the chosen path is one with the best branch metric among possible paths. Otherwise, that is $F_a(l) = 1$, an alternative path is chosen.
(iv) Compare the current path metric with the current threshold $T$. If it is larger than or equal to $T$, go to next step. Otherwise go to step(vii) for backtracking operation.
(v) Tighten threshold. In order to avoid the decoder's infinite searching in backtracking previously examined branches, the threshold must be tightened when the decoder move forward to a certain node for the first time.
(vi) Move forward by increasing $l$ by one. If $l$ is the end of trellis, release the searched path as the correct path. Otherwise go to step (iii).
(vii) Set $F_r(l)$ to 1, which means that threshold violation has happened.
(viii) Move backward to the previous node at depth $l-1$. If $\Lambda(l)$ is less than the current threshold $T$, lower the threshold by $T - \Delta T$ and then go to step (iii). Otherwise go to next step.
(ix) If an alternative branch (or path) exists, go to step (viii). Otherwise go to next step.
(x) Set $F_{j}(l)$ to 1. If $\Delta(l)$ is greater than $T + \Delta T$, go to step (iii). Otherwise reset $F_{j}(l)$ to 0, then go to step (iii).

Some researchers\[^1\][\[^2\]\] have suggested that a *non-sorting* depth-first algorithm is preferred for an efficient sequential algorithm. At this point, the Fano algorithm is a good candidate for an efficient decoding performance. Thus we consider Fano-type decoding methods in this thesis.

Figure 4.2 Fano Algorithm

Figure 4.3 shows the metric behaviour for the correct path and incorrect path. In the figure it is shown that the correct path can be found with the metric linearly increasing at no error occurrence while, with an error occurrence, it can be found after some threshold adjustment. Moreover, we can see that the metric of an incorrect path decreases radically as the depth increases.
4.5 Sequential Decoding for Block Codes

4.5.3. Error Events in Sequential Decoding

In a sequential decoding procedure, decoding failures happen when the decoder is unable to find the correct path. There have been some useful results [2][14][48] for analysis of decoding failure in sequential decoding. In this thesis, we call this decoding failure the decoding error event in sequential decoding and classify possible error events into three cases as follows.

The first error event happens in the case that the decoder has found an incorrect path which is more likely than the transmitted code word. This error event cannot be corrected even by a ML decoder. We call this error event an uncorrectable error event and the probability of this event is represented by $P_r$.

The second event arises from the limitation of computations because of insufficient storage capacity in a practical system. If a large number of computations are required to decode a symbol, the decoder must maintain a large memory to buffer incoming data until the correct path can be found. However, the computations have to be limited inevitably to avoid overflow of the decoding buffer. This amount of computations is called computational limit $L$. We call an error event associated with reaching the computational limit the error event by buffer overflow with probability of $P_r$. In a practical system, this problem can be handled in two ways. One is to deliver just the information symbols in a systematic code or to take linear combinations in a non-systematic code. The other method

Figure 4. 3 Different Behavior of Correct Path and Incorrect Path

Chapter 4. Sequential Decoding Approach for Reed-Solomon Codes
is to erase the affected sequence for retransmission. Actually this type of error is the most serious in most sequential decoders[14].

The last error event occurs in the case that the sequential decoder has found a wrong path as if it were the correct path. This error event may be influenced by the metric ratio, since even a metric ratio giving the most efficient performance cannot always produce the maximum likelihood performance. This point is well explained by Clark and Cain [11] and they suggested a solution that the most cost-effective way to improve this error event is not by changing the metric ratio but by increasing constraint length. This type of error occurrence is called the error event by wrong path with the probability $P_w$. However, this error event would not occur in maximum likelihood decoding such as the Viterbi algorithm which searches for all possible paths.

Those three error events can be used to analyse the decoding performance of a sequential decoder. Figure 4.4 shows the error event diagram. In the figure, we can see that some error events which are regarded as decoder overflow and a wrong path, may be included in the error event of ML decoding. This can be explained by the fact that some decoding errors, classified into those two error events, may be uncorrectable errors even though sufficient computations are given.

![Figure 4.4 Probabilities of Error Events](image)

If the error probability of ML decoding is represented by $P_{ML}$, overall error probability $P_v$ of the sequential decoding has the relationship of

$$P_v \leq P_{ML} \leq P_v = P_o + P_w + P_c$$

The equation 4.13 explains that the performance of the sequential decoding can approach to ML performance by the reduction of the probability of $P_o$ and $P_c$. 

---

*Chapter 4. Sequential Decoding Approach for Reed-Solomon Codes*
4.6. MODIFIED FANO ALGORITHM FOR REED-SOLOMON CODES

As we discussed in the previous section, it is desirable to minimise two error events, buffer overflow and wrong path search, so that the performance of the Fano sequential decoding approaches ML performance. In this section, we present a modified Fano algorithm in which the original Fano algorithm is modified to achieve the improvements of these two error events.

4.6.1. Improvement of Decoding Error Events

The error event by wrong path search can be reduced by checking the path searched. We now introduce a decision rule function to reduce such an error event. The role of this function compares the path metric of a searched path with a threshold which is chosen properly. If a searched path metric is less than the threshold, the path is ignored and the backtracking operation is initiated to search for other paths satisfying the threshold. Since the value of the threshold affects the computation, it is important to choose the appropriate value from the viewpoint of efficiency. If the value is unnecessarily high, it will require more computations without a significant improvement in performance.

Fano[14] proposed a threshold condition shown in equation (18) of his paper and we interpret this for the binary input and Q-ary output channel such that

\[
\sum_{i=1}^{N} \log \frac{P(j|i)}{P(j)} \geq N \cdot R
\]  

(4.14)

\[
\sum_{i=1}^{N} \log \frac{P(j|i) - R}{P(j)} \geq 0.0 = T_p
\]  

(4.15)

where \( P(j) \) is the total probability of observing an output in the j-th quantization level, \( P(j|i) \) is the probability of observing in the j-th quantization interval given that the symbol i is transmitted, \( N \) is the length of the output sequence, and \( R \) is the code rate.
Equation (4.14) can be represented by (4.15). Then we can see that the left term of (4.15) represents the whole path metric. Therefore we can get a useful interpretation that the path metric of the correct path is lower bounded by zero. By this bound condition, we find the decision rule to qualify a searched path as the correct path.

Next we consider the error event by decoder buffer over flow with probability $P_e$. Since sequential decoding is a sub-optimum algorithm that does not search all nodes of the trellis structure, the set of searched paths can be classified into two groups at a trellis level, one group including the correct path and the other group not including the correct path. However, the correct path may be lost due to very low values of path metric in very noisy channel conditions even though the path has been searched already. This gives a critical shortcoming of sequential decoding which can be solved by a path updating function as follows.

In this scheme, the decoder calculates the whole path metric of a searched path at the end of the information sequence, since there is a unique path during the redundancy block of a code word. Whenever the path metric of a searched path is greater than the best one which has been recorded so far, the path is updated as the best path. In the meantime, when the decoder reaches the computational limit without any valid path search satisfying the decision rule, it releases the best path as the decoded one.

---

**Chapter 4. Sequential Decoding Approach for Reed-Solomon Codes**
If the correct path has been tried at least once, the decoder cannot lose the maximum likelihood path. Thus it is important that the searching operation is carried out in the region including the correct path. Furthermore if the computational limitation is properly fixed, the decoder can avoid unnecessary computations under the worst channel condition and overall computations can be reduced considerably as well. Figure 4.6 shows the flow of the path updating function at depth $k$ which is the end of information sequence.

![Figure 4.6 Path update process](image)

### 4.6.2 Modified Fano Algorithm for Reed-Solomon Codes

A modified Fano algorithm (MFA) is presented on the binary trellis for RS codes explained in section 3.3.3 so that bit-level soft decision information can be fully used. The modifications are the incorporation of the decision rule and path update function which have been discussed in the previous section.

The path update function has the role of updating the current path at the end of information block. Since the remaining $m(n-k)$ path is uniquely known by the trellis state obtained at the end of information block, the whole path metric can be calculated at this stage. In the scheme, this sequential decoder will produce ML performance only for a set of paths which the decoder has tried within a computational limit.

The decision rule is used at the end of the trellis to qualify a search path. When the decoder finds a candidate path, this path is regarded a valid path satisfying a condition that its path metric is larger than the path threshold bound $T_p$ shown in equation (4.15). However, since this bound may be too loose to be applied to certain codes, the bias term $B$ is added and the value of $B$ will be decided empirically.
Figure 4.7 Modified Fano Algorithm

Figure 4.7 shows the flow chart of the modified Fano algorithm using binary-branch trellis for $(n, k)$ Reed-Solomon codes over $GF(2^m)$. The numbers of bit constituents of a code word and of the information block are $mn$ and $mk$ respectively. The variables and constants which has been shown in Table 4.2, are also used. The basic decoding operation is similar to the standard Fano algorithm illustrated in Figure 4.2. Comparing to the Figure 4.2, there is the path update function at the end of information block and the decision rule function at the end of whole trellis. If a searched path fails to satisfy the decision rule, a back tracking operation is initiated until a valid path has been found. In very noisy channel conditions, where there is no valid path satisfying the decision rule, the correct path can be found by the path update function only if the path has been tried.
4.6.3. Performance Analysis of Modified Fano Algorithm (MFA)

Simulation is carried out on the simulation test bed which has been explained in section 3.6. As a system model, we use a BPSK system with 8-level quantizer through an AWGN channel. The performance of MFA is analysed from the point of view of complexity and error-correcting performance. The complexity is measured by the average computations per information bit and the error-correcting performance represented by bit error rate $P_e$.

4.6.3.1. Optimization of Decoding Variables

The variables of Fano sequential decoding affect the efficient decoding operation. In particular, the threshold spacing $\Delta T$ and computational limit $L$ must be optimized for cost-effective performance with a given code. Thus the sensitivities according to $\Delta T$ and $L$ are observed in the following experiments.

4.6.3.2. Sensitivity of $\Delta T$

The threshold spacing value $\Delta T$ is one of the important parameters for proper and efficient behavior of Fano algorithm. (Lin and Costello [31] showed a guideline that $\Delta T$ should be between 2 and 8 for unscaled metrics.) In order to inspect the sensitivity of $\Delta T$, four different values of $\Delta T$, 2.0, 3.0, 4.0, and 5.0 are used for standard Fano sequential decoder. Each bit error rate with respect to three error events is measured at $\text{Eb/No}=4.0$ for (7,3) and (15,9) RS codes. The values of $P_r$, $P_w$, $P_e$, and $P_e$ represents the bit error rate of uncorrectable errors, a wrong path search, decoder buffer overflow, and overall decoding error rate respectively. The computational limit are fixed at $2^5$ and $2^8$ for (7,3) and (15,9) RS codes.

Figure 4.8 and Figure 4.9 demonstrate that as $\Delta T$ is small, the error events by decoding buffer overflow increased, while the error event by wrong path decreased. This phenomenon is explained by the fact that too small $\Delta T$ caused premature backtracking operations and too large $\Delta T$ allowed many incorrect paths.
Table 4.3 shows the decoding complexity according to different values of $\Delta T$. Considering the decoding performance results, we can see that the choices of $\Delta T = 4.0$ for (7,3) RS codes and $\Delta T = 3.0$ for (15,9) RS codes seem reasonable, since they produced low decoding error rate at the proper complexity cost.
### Table 4.3 \( \tilde{C} \) for (7,3) and (15,9) RS codes at \( L = 2^8 \)

<table>
<thead>
<tr>
<th>( \Delta T )</th>
<th>(7,3) RS codes</th>
<th>(15,9) RS codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0</td>
<td>5.73</td>
<td>181.62</td>
</tr>
<tr>
<td>3.0</td>
<td>5.01</td>
<td>179.67</td>
</tr>
<tr>
<td>4.0</td>
<td>4.44</td>
<td>179.57</td>
</tr>
<tr>
<td>5.0</td>
<td>4.18</td>
<td>200.02</td>
</tr>
</tbody>
</table>

#### 4.6.3.3. Sensitivity of Computational Limit \( L \)

Computational limit \( L \) exists to avoid decoder buffer overflow. A large value of \( L \) contributes to low decoding error rate by giving enough computations so that the decoder can search for the correct path. However, for the efficiency of computations, this value will be optimised according to the number of possible trellis states at a given code. Thus the sensitivity to computational limit should be considered.

An experiment is carried out for different values of \( L \) for (7,3) and (7,5) RS codes. Figure 4.10 and Figure 4.11 are the decoding performance and the complexity for those RS codes.

In Figure 4.10, for (7,3) RS codes the choice of \( L = 512 \) is regarded as the optimised value, since \( L = 768 \) gives almost the same decoding performance and complexity as for \( L = 1024 \). In the same way, we can see that \( L = 512 \) is the optimised value for (7,5) RS codes. This means that those chosen values are sufficient to search for the correct path at these RS codes.

In Figure 4.11 it is verified that further increasing the computational limit does not influence the decoding complexity since sufficient computation has been allowed already. Moreover we can see that the decoding complexity of (7,3) RS codes is higher than that of (7,5) RS codes when the same computational limit is given. This result is explained by the fact that the number of possible trellis states for (7,3) RS codes is larger than that for (7,5) RS codes. Thus as the code length is large with large Galois fields, the value of the computational limit required will increase enormously. In such case, it is difficult to allow a sufficient computational limit and the error event by decoding overflow will dominate overall decoding error events. In order to achieve near-ML
decoding performance at a reasonable computational cost, it is necessary to solve this problem.
4.6.3.4. Decoding Error Probability of MFA

In this section, the comparison between the standard Fano algorithm (SFA) and the modified Fano algorithm (MFA) is discussed using the simulation results for (15,9) RS codes. The decoding parameters \( L \) and \( \Delta T \) were given with the optimised values.

As we described before, MFA has two additional functions, path update function and decision rule function, to improve the error correcting probability of SFA. In order to verify the effect of the two modifications, we compared the four simulation results for (15,9) RS codes, which are shown in Figure 4.12, Figure 4.13, Figure 4.14, and Figure 4.15.

Figure 4.12 is the result for SFA, which does not have the two modifications. In the figure, error event by the wrong path becomes more serious as \( E_b/N_0 \) increases and the error event by decoding overflow occupies a large portion of overall decoding error probability \( P_e \).

Figure 4.13 is the result which shows the effect of the path update function using only the path update function without the decision rule. Compared with Figure 4.12, the probability \( P_e \) has been reduced and \( P_w \) increases instead, while \( P_e \) and \( P_w \) is not changed. This shows that some errors, which were included in \( P_e \), have been moved to \( P_w \) by the path update function, since they were uncorrectable errors.

Figure 4.14 shows the effect of the decision rule function by using only the decision rule without the path update function. Again comparing with Figure 4.12 and Figure 4.13 the probability \( P_w \) has been reduced considerably, and \( P_e \) and \( P_w \) increases instead. This shows that the use of the decision rule improves the error events by wrong path search but the decoder has to spend more computations because of the back tracking initiation to qualify the paths according to the decision rule. Moreover we can see that the decision rule without the path updating function can lose the correct path when decoding overflow happens earlier than in SFA because of the increased computations.
Figure 4.15 is the result of MFA, which employs the path updating and decision rule function. It is shown that the probabilities of $P_u$ and $P_e$ are reduced and thus overall decoding probability $P_e$ has been improved. Comparing with SFA, MFA produced around 0.4 dB gain at $10^{-4}$ BER. It is shown that as Eb/No increases, the gain obtained grows. This gain resulted from the modifications, since a searched path of low path metric has been rejected for a better path by the decision rule, while the most likely path can be found by the path update function at decoder over flow.

![Decoding Error Probability for (15,9) RS codes](image)

Figure 4.12 Decoding Error Probability of standard Fano algorithm for (15,9) RS codes
4.6 Modified Fano Algorithm for Reed-Solomon Codes

Figure 4.13 Decoding Error Probability of Fano algorithm with Path Update Function for (15,9) RS codes

Figure 4.14 Decoding Error Probability of Fano algorithm with Decision Rule for (15,9) RS codes
4.6 Modified Fano Algorithm for Reed-Solomon Codes

Decoding Error Probability for (15,9) RS codes

Modified Fano Algorithm

![Decoding Error Probability Graph](image)

Figure 4.15 Decoding Error Probability of Modified Fano algorithm with Path Update Function and Decision Rule for (15,9) RS codes

4.6.3.5. Complexity of MFA

Table 4.4 is the complexity comparison between the standard Fano algorithm (SFA) and the modified Fano algorithm (MFA). The complexity is measured by $\bar{C}$, the average computations per information bit. In the table, we can see that additional computations are required because of the decision rule function. It is also shown that this additional cost decreases considerably as $Eb/No$ increases.

<table>
<thead>
<tr>
<th>$\frac{E_b}{N_o}$ (dB)</th>
<th>$\bar{C}$ at SFA</th>
<th>$\bar{C}$ at SFA with path updating function</th>
<th>$\bar{C}$ at SFA with decision rule function</th>
<th>$\bar{C}$ at MFA</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>583.99</td>
<td>583.99</td>
<td>963.46</td>
<td>963.46</td>
</tr>
<tr>
<td>4.0</td>
<td>175.41</td>
<td>175.41</td>
<td>261.45</td>
<td>261.45</td>
</tr>
<tr>
<td>5.0</td>
<td>45.75</td>
<td>45.75</td>
<td>56.81</td>
<td>56.81</td>
</tr>
</tbody>
</table>

Table 4.4 Complexity Comparison for (15,9) RS codes

Chapter 4. Sequential Decoding Approach for Reed-Solomon Codes
4.6.3.6. Tightened Decision Rule of MFA

The path threshold $T_p$ used in the decision rule function in MFA can be too loose to minimize the error event by wrong path search. The reason is that the action of the decoder is carried out only on the information block and a long parity check block can give rise to a close sequence but incorrect path satisfying $T_p$. Therefore the bias term $B$ is required to lower the decoding error probability by wrong path search.

Table 4.5 shows the sensitivity by different bias $B$ for (15,9) RS codes at Eb/No=4.0 dB. In the table, it is shown that a large value of $B$ contributes to reducing the probability $P_w$ by wrong path search but the complexity, which is represented by $C$ (average computations per information bit) increases. Moreover the probability $P_e$ is not affected by the variation of $B$, while the probability $P_r$ is increased as $B$ is large. It means that some errors which have been included in the event by wrong path search, cannot be corrected, since the searched paths are the most likely paths.

<table>
<thead>
<tr>
<th>$B$</th>
<th>$P_r$</th>
<th>$P_w$</th>
<th>$P_e$</th>
<th>$P_r$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0</td>
<td>0.0023</td>
<td>0.00058</td>
<td>0.003</td>
<td>0.0059</td>
<td>987.99</td>
</tr>
<tr>
<td>3.0</td>
<td>0.0024</td>
<td>0.0001</td>
<td>0.003</td>
<td>0.0055</td>
<td>1381.20</td>
</tr>
<tr>
<td>6.0</td>
<td>0.0024</td>
<td>0.0</td>
<td>0.003</td>
<td>0.0054</td>
<td>2022.87</td>
</tr>
<tr>
<td>9.0</td>
<td>0.0024</td>
<td>0.0</td>
<td>0.003</td>
<td>0.0054</td>
<td>30000.49</td>
</tr>
</tbody>
</table>

From the point of view of complexity, if the bias $B$ is too high, the computations may increase unnecessarily, since even the correct path found can be regarded as a wrong path and can initiate the back tracking operation. Thus it is important to take the best value. As the redundant block (parity check block) of a code word is large, the value of bias term $B$ will be higher.
4.7. DISCUSSION

In this chapter, a modified soft-decision sequential decoding has been introduced for RS codes using bit-level soft-decision information. This new approach has the following noticeable features.

First is the complete implementation of bit-level channel measurement information (soft-decision). This can be achieved by the binary-branch trellis obtained from the binary systematic parity check matrix. The use of bit confidence will bring even more significant improvement to the error correcting performance and complexity in the probabilistic decoding methods using the sequential decoding algorithm, since the decoder can find the most likely path at the least cost.

The second feature is the complexity reduction of the Fano sequential algorithm. A powerful feature of the Fano algorithm is the backtracking operation to retrace the current path on the detection of a wrong one and search for an alternative. The efficiency of the backtracking operation significantly affects the overall complexity. The binary-trellis structure provides a more efficient alternative searching by allowing only single alternative searching at each node. Although the binary-trellis has greater depth than a non-binary trellis, since the complexity of the sequential algorithm increases linearly with trellis depth and exponentially with trellis branch size, this method will be very effective in the application to large Galois fields.

The modified Fano algorithm produces the most likely path at a given computational limit. The most likely path can survive through the path updating function. Moreover, the decision rule contributes to blocking the selection of a wrong path which has been made at the end of trellis. Thus if a searching operation has been in a region which is likely to contain the correct path, the decoding performance will approach that of ML decoding.

From the point of view of decoding performance and complexity, it is important to optimise certain parameters of the modified Fano sequential decoding. For best operation of the sequential decoder, threshold space value $\Delta T$, computational Limit $L$, and path threshold bias $B$ are optimised. In particular the parameter $\Delta T$ depends on the capacity of the channel.
computational limit \( L \) and the desired decoding performance. If sufficient computations are allowed, too large \( \Delta T \) may be more likely to give a wrong path because the threshold is loose. On the other hand, too small \( \Delta T \) may force the decoder to do many computations and prevent the decoder from arriving at the end of the trellis when insufficient computation limit is allowed. Thus the optimum \( \Delta T \) will be carefully decided with other decoding parameters.

From the simulation results in the previous sections, we can verify that the computational limit \( L \) influences the reduction of the error events caused by decoding over flow, while the bias \( B \) is related to the improvement of the events caused by wrong path search. The best choices of those parameters depend on the code length and code rate of a given code word as follows.

For a code over a given field, a code with low rate requires a larger computational limit \( L \) and value of bias \( B \) than one with high rate, since the number of trellis states increases and the bias term given by code rate \( R \) is relatively small.

As a conclusion, one important issue in using this new sequential decoding is how the searching region can be directed to the most likely region. If the decoder can always follow the shortest way to arrive at the correct path within a limit of computation, it is ideal for obtaining the most cost-effective performance. In the next chapter, some techniques for complexity reduction will introduced.
CHAPTER 5

COMPLEXITY REDUCTION USING PERMUTATION DECODING

5.1 INTRODUCTION

In the previous chapter there was discussion of bit-level soft-decision sequential decoding using a modified Fano algorithm (MFA) for RS codes. This approach has many advantages over previous methods from the point of view of decoding performance and complexity. However, the decoding complexity is still a critical restriction for the application to long RS codes. In this chapter, we present some methods to reduce the decoding complexity without loss of error correcting probability.

Firstly, some experiments are carried out to estimate the complexity of a sequence for sequential decoding. In these experiments, computational variations are obtained according to several error patterns and then we suggest a guide for convenient sequence selection which is useful for design of low complexity sequential decoding.

Next we introduce the concept of permutation decoding for reducing the decoding complexity. The basic idea of this approach is to maintain a single trellis structure for the code and to implement the convenient sequence-first search using permutation groups of RS codes. For this, we use a criterion which represents the information block confidence in a sequence. As a permutation technique, cyclic permutation is presented using the cyclic property of RS codes on a symbol basis. Then two kinds of cyclic permutation decoding schemes are described. Through simulation, their performance is demonstrated in terms of decoding error rate and complexity compared with non permutation decoding.
5.2 COMPLEXITY ESTIMATION OF SEQUENTIAL DECODING

In spite of a random property in the decoding complexity of sequential decoding, it is well known that the complexity characteristic depends on the number of errors and the error location[11][38].

In this section, the characteristic of sequential decoding is examined by experiments to see the complexity variation according to three factors; error location, burst length, and confidence of error bit. For these experiments, the error patterns associated with the three factors are generated and then the computations required to decode correctly are measured.

5.2.1 Computational Variation according to Error Location

Computation variation according to error location is examined. For \((n, k)\) RS codes over \(GF(2^m)\), \(m \cdot n\) sequences with a single error at different bit positions, are given, and then computations required for correct decoding are calculated in each case. Thus 21, 60 and 155 sequences are used for \((7,3)\), \((15,9)\), and \((31,27)\) RS codes respectively.

Figure 5.1 Computational Variation according to Error Location

Figure 5.1 shows the simulation results. The horizontal axis indicates the location of the single error bit and the vertical axis is the number of computations required to
decode the single error bit correctly. In the figure, it is seen that the number of computations rapidly decreases as the error bit is located in a later position of the $m \cdot k$ information block, while the number of computations associated with errors in the $m \cdot (n - k)$ parity check block is relatively less dependent on position. In addition, the computational difference among $m$ bits constituting a symbol is relatively trivial.

5.2.2 Computational Variation according to Spread Width of Error Bits

We examine another computational variation according to the spread width of error bits in a code word. For this experiment, we generate the following five types of error patterns for (15,9) RS codes;

- Type 1 is an error pattern which has three consecutive error bits.
- Type 2 is an error pattern having three consecutive single bit symbol errors with the same bit wrong in each case.

We assume that a symbol error has a single error bit in the following cases too.

- Type 3 is an error pattern which has three symbol errors with one symbol interval.
- Type 4 is an error pattern which has three symbol errors with two symbol interval.
- Type 5 is an error pattern which has three symbol errors with three symbol interval.

![Figure 5.2 Error Spread Width for (15,9) RS codes](image)

Figure 5.2 illustrates the received sequence made by five types of error patterns. In the figure, a small shadow circle indicates an error bit while a large shadow circle is an error symbol.
5.2.3 Computational Variation according to Confidence of Error Bit

In definition 3.8, we introduced the concept of bit confidence corresponding to its soft decision level. In this section, the computational variation according to the different confidence levels is obtained. The error pattern is set up in similar way to the experiment in 5.2.1, but each error bit has different confidence: 0, 3, 6, and 7 with
5.2 Complexity Estimation of Sequential Decoding

respect to 16-level soft-decision values. The horizontal axis indicates the location of a single error bit and vertical axis is the number of computations.

In Figure 5.4, it is shown that when an error bit has a high confidence value, the computations required increase. This can be explained by the fact that the error bit with high confidence makes the decoder move further along a wrong path because of relatively large bit metric and thus more computations are needed to return to the correct path by back tracking operations.

![Computation Variation by Error Bit Confidence](image)

Figure 5.4 Complexity Variation by Different Confidence of an Error Bit

5.2.4 Discussion

In the experiments, we have seen that error location, error spread width, and confidence of error bit influence the complexity of the sequential decoding. However, the error pattern used in the experiments may not reflect all phenomena obtained from normal channel behavior due to a number of noise sources.

In spite of such difficulty, with the result obtained from the three experiments, it is possible to derive a useful guide so that the decoder can estimate before decoding the complexity of the received sequence including a certain error pattern. This is called the Guide to Convenient Sequential Decoding (GCSD) as follows.
(i) The location of error bit is the most critical factor in complexity of the sequential decoder. The computation variation by different locations of an error bit within a symbol, is relatively trivial compared with that of different symbol locations.

(ii) As the number of errors in the information block increases, the required computations increase.

(iii) As the spread width of errors in a code word is close (dense), the decoder requires more computations than in the case of sparse errors.

When the demodulator provides soft-decision information as an input to the decoder, the confidences of the received bits can be statistically estimated as we have verified in chapter 3.7. Thus the complexity of a sequence for sequential decoding can also be estimated in the case of GCSD. In a sequential decoding operation, if the decoder can avoid an undesirable error pattern as GCSD indicates, it will perform a low complexity decoding. In the next section, we explain some methods to achieve this.

5.3 PERMUTATION DECODING

An effective strategy for low complexity sequential decoding is for the decoder to face the most convenient sequence following GCSD (Guide to Convenient Sequential Decoding) as discussed in the previous section. Thus if a received sequence can be changed into a certain equivalent one which has a less complex error pattern, the decoder would search for the correct path with lower computation.

At this point, a permutation group of RS codes can be a good solution for providing many equivalent code words so that the decoder can choose the most convenient one among the group. Each element of this group must keep the same property of weight distribution and minimum distance. When a received sequence contains some error, a permutation of the sequence may give the desirable effect that the location of error bits is also changed. An error pattern in a received sequence is changed with the permutations so that each permuted sequence has a different decoding complexity.
Thus the decoder can perform a convenient sequence-first sequential decoding by choosing the most convenient one among the group under the consideration of GCSD.

In this thesis, we will introduce three types of permutation techniques; Cyclic permutation, Squaring-permutation, and Hybrid-permutation. In the rest of this chapter, cyclic permutation of RS codes is discussed to implement low complexity sequential decoding.

5.3.1 Cyclic Permutation of Reed-Solomon Codes

We consider \((n, k)\) Reed-Solomon codes over \(GF(2^m)\). Let a RS code word denote \(c(x)\) as

\[
c(x) = \sum_{i=0}^{n-1} \alpha_i \cdot x^i \quad \text{for} \quad \alpha_i \in GF(2^m).
\]  

(5.1)

The \(n\) cyclic permutation group \(G_c(c(x))\) is defined as

\[
G_c(c(x)) = \left\{ \sum_{i=0}^{n-1} \alpha_i \cdot x^{i+\beta} \right\} \mod (x^n - 1)
\]  

(5.2)

where \(\beta \in \{0,1,2,\cdots,n-1\}\). The cyclic permutation \(\pi_c(\beta)\) for any \(\beta\) can be represented by

\[
\pi_c(\beta) \cdot c(x) = \sum_{j=0}^{n-1} \alpha_{(j+\beta) \mod n} \cdot x^j
\]  

(5.3)

By cyclic permutation of an \((n, k)\) RS code over \(GF(2^m)\), we can get \(n\) different sequences which are also a code word. If this cyclic permutation is carried out on a received sequence, error locations and confidences of the sequence are preserved relative to the cyclically shifted code word. Thus any sequence among the permutations can be simply applied to some trellis using the modified Fano sequential decoder which has been described in chapter 4.

Example: Consider a \((7,3)\) RS code word over \(GF(2^m)\) represented by

\[
c(x) = \alpha^5 + x^1 + \alpha^6 x^2 + \alpha^2 x^3 + \alpha^3 x^4 + 0 x^5 + \alpha x^6
\]

The cyclic permutation group is obtained as
\[ \pi_6(6) \cdot c(x) = \alpha + \alpha^2 x + x^2 + \alpha^6 x^3 + \alpha^2 x^4 + \alpha^3 x^5 + 0, \]
\[ \pi_5(5) \cdot c(x) = 0 + \alpha x + \alpha^5 x^2 + x^3 + \alpha^6 x^4 + \alpha^2 x^5 + \alpha^3 x^6, \]
\[ \vdots \]
\[ \pi_1(1) \cdot c(x) = 1 + \alpha^6 x + \alpha^2 x^2 + \alpha^3 x^3 + 0 + \alpha x + \alpha^5 x^2. \]

If each of these sequences is assigned a priority under the consideration of GCSD before the decoding, the decoder can start to decode in order of convenient sequence for which the error pattern is likely to give low complexity. This decoding method is called cyclic permutation sequential decoding (CPSD). In this method, since cyclic permutation of Reed-Solomon codes is carried out symbol-by-symbol, although the decoder uses the bit level confidence during decoding operation, the confidence of each symbol must be considered according to definition 3.9.

By the definition 3.10, when \( n \) cyclic permuted sequences are obtained from a received sequence \( S \), \( I_i \) denotes Information Block Confidence (IBC) for a cyclic permuted sequence \( \pi_c(i) \cdot S \) such that

\[ I_i = \sum_{j=0}^{k-1} f_{(i+j) \mod n} \quad \text{for } i = 0, 1, \ldots, n - 1 \]  \hspace{1cm} (5.4)

Since the sequence with the largest IBC will statistically contain the most reliable information block, it can be estimated as the most convenient one, which is likely to contain the smallest number of errors in the information block. If \( n \) candidate sequences can be assigned their priorities in order of the values of IBC, the sequential decoder can effectively search for the correct path on a convenient sequence-first search principle. However, we can employ a different estimate of complexity, such as a smaller section of the information block if necessary. This point will be discussed later.

### 5.3.2 Cyclic Permutation Sequential Decoding (CPSD)

The sequential decoder using cyclic permutation is presented in this section. As a decoding algorithm, the Modified Fano algorithm in chapter 4 is employed and IBC is used as the criterion to choose the most convenient sequence among the cyclic permutation group before decoding.

---

Chapter 5. Complexity Reduction using Permutation Decoding
First of all, each permuted sequence is assigned its priority in order of IBC of the sequence. Since the number of errors of information block is the most important factor which influences the decoding complexity, it is predicted that a sequence with high IBC has statistically low error occurrence in information block. Figure 5.5 shows the process to select the most reliable information sequence at the received sequence. Once the priorities have been decided, the decoder can start to decode from the sequence with highest priority.

![Figure 5.5 Sequence Sorting in Cyclic Permutation](image)

The cyclic decoder can be classified according to the number of sequences tried by the decoder. One possibility is that the decoder uses a single sequence with the largest IBC. It is called as Single Cyclic Permutation Sequential Decoder (SCPSD). The other is that the decoder uses several sequences in order of priority instead of a single sequence as in SCPSD. This decoding method is referred as Multi-sequence Cyclic Permutation Sequential Decoding (MCPSD).

With severe noise, a specially complex error pattern cannot be managed by a single sequence, since the choice of one single sequence by IBC may not be the best one because of different confidences of error bits. In this case, the decoder may waste decoding time along a repeated path without finding any valid path. Thus the use of a single sequence may not be enough and further sequences with lower priorities can be helpful to search for the correct path.

Figure 5.6 shows that for a special error pattern, the use of multi-sequences can be more helpful than a single sequence. If confidences $C_4$ and $C_5$ are lower than $C_{13}$, $C_{14}$,
the sequence $S_1$ is the best sequence with the largest IBC, although actually the sequence $S_2$ is more convenient in the decoding operation since the combination of error location is better than the sequence $S_1$.

Figure 5.6 Multi-sequences Use

In the MCPSD scheme, there are two more decoding parameters, $Y$ and $L_c$ where $Y$ is the number of sequences tried and $L_c$ is the computational limit per sequence. Since MCPSD has at least the same computational limit $L$ as that of SCPSD, $L_c$ and $Y$ are chosen with the relationship as

$$L \geq L_c \cdot Y$$  \hspace{1cm} (5.5)$$

where $L$ is overall computational limit. The number of sequences to be tried can be empirically determined according to a given code. This will be shown later.

Figure 5.7 shows the cyclic permutation decoding procedure for SCPSD and MCPD. Firstly the overall computational limit, $L$, and the number of trial sequences, $Y$, are decided. If a single sequence is chosen, $Y = 1$ and the decoder is SCPSD, otherwise MCPSD is performed. Once the candidate with the largest IBC in the cyclic permutation group has been chosen, the sequence is decoded by the modified Fano algorithm (MFA) which has been introduced in chapter 3. The MFA provides two types of decoding result. If a valid path satisfying a given decision rule has been found, the decoder successfully releases the path as a correct decoded one. On the other hand if the decoder reaches a given computational limit without finding a valid path, the best sequence is released by path update function. Moreover when the decoding is stopped by the computational limit, MCPSD tries another candidate with
5.3 Permutation Decoding

next priority within a given maximum number of trials, \( Y \). At the worst case when no valid path has been found even at the maximum number of trials, the decoder releases the best path with the largest path metric amongst paths which have been obtained at each trial. Once the searching procedure for a received sequence ends, by restoring the order of the released decoded code word, a final decoded word is obtained.

![Flowchart of Cyclic Permutation Sequential Decoding](image)

Figure 5.7 Cyclic Permutation Sequential Decoding

5.3.3 Simulation Result in Cyclic Permutation Sequential Decoding (**CPSD**)  
Simulation is carried out using BPSK modulation with 8-level quantisation scheme over a Gaussian channel. Complexity is measured by average computations per bit,
and error correcting performance is calculated by bit error rate (BER) with respect to Eb/No.

5.3.3.1 Error Location Distribution by Cyclic Permutation

In order to see the effect of cyclic permutation, we have an experiment for (7,5) and (15,9) RS codes. For the experiment, the best sequence with the largest IBC amongst cyclic permutation group for a received is chosen and the bit error rate is measured for each bit position. Simulation is carried out at Eb/No=4.0 dB using 8-level quantisation scheme.

Figure 5.8 and 5.9 show that the bit error rate in choosing the sequence with the best IBC, is lowered by cyclic permutation while for non permutation it is constant. This means that the errors in the information block can be moved to the parity check block by choosing the sequence with the largest IBC amongst the cyclic permutation group. This error relocation will provide a more convenient sequence for sequential decoding according to GCSD. Moreover we can see that the use of IBC is effective to reduce the error occurrence in the first part of the sequence, which is undesirable for efficient sequential searching.

This error relocation by cyclic permutation will be useful to reduce complexity for a decoding scheme which error location is critical factor in decoding complexity. Moreover there is possibility for improving error correcting performance by avoiding a worst case in which a decoding algorithm cannot manage effectively. Therefore when a permutation group is provided, it is an important task to find a criterion for the convenient sequence with respect to a decoding algorithm.
5.3 Permutation Decoding

Error Relocation by Cyclic Permutation for (7,5) RS codes
at 4.0 dB with Q=8

Figure 5.8 Error Location Distribution by Cyclic Permutation

Error Relocation by Cyclic Permutation for (15,9) RS codes
at 4.0 dB with Q=8

Figure 5.9 Error Location Distribution by Cyclic Permutation

Chapter 5. Complexity Reduction using Permutation Decoding
5.3.3.2 Convenient Sequence Estimation by IBC

We examine if IBC is a proper criterion to decide the most convenient sequence among possible candidates of a permutation group. However, there may be other choices for deciding convenient sequence instead of IBC. For this, we take different confidence measurements corresponding to one symbol, half of the information block size (1/2 IBC), and the whole of the information block (IBC), to decide the most convenient sequence.

Table 5.1 and Table 5.2 shows the simulation results for (15,9), (15,11) and (15,13) RS codes at Eb/No=4.0 and 5.0 dB. In the results, we can see that the best value of BERs are obtained from using IBC for (15,9) and (15,11) RS codes while for (15,13) RS codes, the values of BER are almost same with the different confidence measurement blocks. On the other hand, in the aspect of the complexity which is measured by $\bar{C}$ (average computations per information bit), the use of IBC provides the lowest complexity and its amount achieved is relatively efficient for RS codes with low rate. Consequently it is verified that since the number of errors within information block is the most important factor, IBC is a good criterion for estimating convenient sequence and it is more effective in the application to RS codes with low code rate. Furthermore this criterion can be used in predicting the decoding complexity among possible candidates in not only cyclic permutation group but also other permutation group.

Table 5.1 Simulation Result for different confidence block size at Eb/No=4.0 dB

<table>
<thead>
<tr>
<th>RS Code</th>
<th>one symbol</th>
<th>1/2 IBC</th>
<th>IBC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BER</td>
<td>$\bar{C}$</td>
<td>BER</td>
</tr>
<tr>
<td>(15,9)</td>
<td>0.00007</td>
<td>795.6</td>
<td>0.0049</td>
</tr>
<tr>
<td>(15,11)</td>
<td>0.0013</td>
<td>152.0</td>
<td>0.0010</td>
</tr>
<tr>
<td>(15,13)</td>
<td>0.0038</td>
<td>66.2</td>
<td>0.0037</td>
</tr>
</tbody>
</table>

Table 5.2 Simulation Result for different confidence block size at Eb/No=5.0 dB

<table>
<thead>
<tr>
<th>RS Code</th>
<th>one symbol</th>
<th>1/2 IBC</th>
<th>IBC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BER</td>
<td>$\bar{C}$</td>
<td>BER</td>
</tr>
<tr>
<td>(15,9)</td>
<td>0.000038</td>
<td>170.8</td>
<td>0.000026</td>
</tr>
<tr>
<td>(15,11)</td>
<td>0.00016</td>
<td>38.01</td>
<td>0.00015</td>
</tr>
<tr>
<td>(15,13)</td>
<td>0.00082</td>
<td>20.2</td>
<td>0.00084</td>
</tr>
</tbody>
</table>
5.3.3.3 Performance Analysis in SCPSD

The performance of single cyclic permutation sequential decoder (SCPSD) is compared with the decoder which does not employ a permutation technique, namely called non permutation sequential decoding (NPSD). The comparison is carried out from the view point of decoding error probability and complexity. Simulation is carried out for (7,3), (15,9) and (31,27) RS codes. The decoding parameters which have been used are shown in Table 5.3. These parameters have been empirically chosen to minimise the two types of error events, namely wrong path search and decoding overflow, (in section 4.5.3) within limits of computational cost.

|                    | (7,3) RS codes | (15,9) RS codes | (31,27) RS codes |
|--------------------|----------------|----------------|
| ΔT                 | 4.0            | 2.0            | 3.0              |
| L                  | $2^9$          | $2^{18}$       | $2^{21}$         |
| B                  | 0.0            | 7.0            | 6.0              |

In Figure 5.10 and Figure 5.11, the simulation results for (7,3), (15,9), and (31,27) RS codes are shown with the comparison between SCPSD (single cyclic permutation sequential decoding) and NPSD (non-permutation sequential decoding). SCPSD with the decoding parameters as shown in Table 5.3 was used in the simulation.

In Figure 5.10, we see that SCPSD produces almost same performance as NPSD for (7,3) RS codes while it gives an improvement of about 0.4 dB at a BER of $10^{-5}$ for (15,9) and (31,27) RS codes. Moreover, the performance gap widens at lower BER’s. These results can be explained by the fact that the computational limit $L$ used is sufficient for (7,3) RS codes, while it is not sufficient for (15,9) and (31,27) RS codes because of a large number of trellis states. There are respectively $2^{24}$ and $2^{20}$ trellis states for the (15,9) RS code and (31,27) RS code compared with $2^9$ trellis states for the (7,3) RS code.

Figure 5.11 is the comparison of complexity between NPSD and SCPSD for (7,3), (15,9), and (31,27) RS codes. It is shown that the complexity of SCPSD is slightly lower, and the complexity gap increases at higher Eb/No for longer codes. This explains the fact that cyclic permutation contributes to providing a more likely sequence and improving the decoding performance when the computational limit is
the same as for NPSD for those longer codes. Note that for (7,3) RS codes, there is no advantage in SCPSD, since the computational limit used was already enough and this code does not have a severe problem of decoding complexity.

Figure 5.10 Decoding Performance Comparison between SCPSD and NPSD

Figure 5.11 Complexity Comparison between SCPSD and NPSD

Chapter 5. Complexity Reduction using Permutation Decoding
5.3 Permutation Decoding

5.3.3.4 Performance Analysis in MCPSD

In a similar way to the previous simulation of SCPSD, the performance of multi-cyclic permutation decoder (MCPSD) is shown Figure 5.12 and Figure 5.13.

5.3.3.4.1 Optimised Sequence Number for MCPSD

In the MCPSD, the choice of maximum-trial-sequence value \( Y \) is important to achieve the most efficient decoding performance. In order to decide the optimised \( Y \) for some RS codes, at a fixed overall computational limit \( L \), the decoding performance and complexity were examined according to different values of \( Y \). At each sequence, the computational limit is determined by \( L_c \) as in equation (5.5).

Table 5.4 contains the results for (7,3) RS codes obtained at \( \text{Eb/No}=3.0 \) and 4.0 dB. In these results, the use of the two best sequences is the most efficient, since it gives significant complexity reduction almost without decoding error probability degradation. Similarly Table 5.5 and Table 5.6 show that the optimal values of \( Y \) are 8 and 16 for (15,9) and (31,27) RS codes.

The results of Table 5.4, Table 5.5 and Table 5.6 show that the use of several sequences with small computational limit \( L_c \) is the most effective for long RS codes. It appears from the experiment that attacking several sequences is more effective in reducing computations rather than of one single sequence with a large computation limit. Thus the decision parameter \( Y \) must be chosen from the viewpoint both of decoding error rate and complexity.

### Table 5.4 Decoding Results for (7,3) RS codes according to different \( Y \)

<table>
<thead>
<tr>
<th>( Y )</th>
<th>( L_c )</th>
<th>( P_e )</th>
<th>( C )</th>
<th>( P_e )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 2^9 )</td>
<td>0.0085</td>
<td>13.4</td>
<td>0.0025</td>
<td>7.38</td>
</tr>
<tr>
<td>2*</td>
<td>( 2^8 )</td>
<td>0.0081</td>
<td>12.71</td>
<td>0.0025</td>
<td>7.16</td>
</tr>
<tr>
<td>4</td>
<td>( 2^7 )</td>
<td>0.0092</td>
<td>12.55</td>
<td>0.0035</td>
<td>7.31</td>
</tr>
</tbody>
</table>

* indicates the optimised \( Y \).
5.3 Permutation Decoding

### Table 5.5 Decoding Results for (15,9) RS codes according to different $Y$

<table>
<thead>
<tr>
<th>$\Delta T = 3.0, B=7.0$</th>
<th>Eb/No=3.0 dB</th>
<th>Eb/No=4.0 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>$L_y$</td>
<td>$P_e$</td>
</tr>
<tr>
<td>1</td>
<td>$2^{18}$</td>
<td>0.0039</td>
</tr>
<tr>
<td>2</td>
<td>$2^{17}$</td>
<td>0.0040</td>
</tr>
<tr>
<td>4</td>
<td>$2^{16}$</td>
<td>0.0039</td>
</tr>
<tr>
<td>8*</td>
<td>$2^{15}$</td>
<td>0.0029</td>
</tr>
</tbody>
</table>

* indicates the optimised $Y$.

### Table 5.6 Decoding Results for (31,27) RS codes according to different $Y$

<table>
<thead>
<tr>
<th>$\Delta T = 5.0, B=6.0$</th>
<th>Eb/No=5.0 dB</th>
</tr>
</thead>
<tbody>
<tr>
<td>$Y$</td>
<td>$L_y$</td>
</tr>
<tr>
<td>1</td>
<td>$2^{21}$</td>
</tr>
<tr>
<td>2</td>
<td>$2^{20}$</td>
</tr>
<tr>
<td>4</td>
<td>$2^{19}$</td>
</tr>
<tr>
<td>8</td>
<td>$2^{18}$</td>
</tr>
<tr>
<td>16*</td>
<td>$2^{17}$</td>
</tr>
</tbody>
</table>

* indicates the optimised $Y$.

5.3.3.4.2 Simulation Results in MCPSD

In Figure 5.12 and Figure 5.13, the simulation results for (7,3), (15,9), and (31,27) RS codes are shown with the comparison between MCPSD and SCPSD. Figure 5.12 shows that MCPSD shows slightly better gain with (7,3) and (15,9) RS codes than SCPSD, while about 0.7 dB extra gain is achieved at a BER of $10^{-5}$ with (31,27) RS codes. On the other hand, in the complexity comparison, MCPSD produced much lower complexity compared with SCPSD for (15,9) and (31,27) RS codes, while the complexity is almost same as SCPSD for (7,3) RS codes. In particular, since MCPSD has provided 0.7 dB gain in Figure 5.12, the complexity reduction obtained is even more significant.

With these results, it is verified that the use of several sequences, instead of one single sequence as in SCPSD, contributes to considerable complexity reduction without decoding performance loss. When a large number of computations are required for the decoding of long RS codes with large field, a single sequence chosen as the most convenient sequence can still require an enormous amount of computation in a severely noisy channel. In such a case, MCPSD performs a more efficient search and

---

Chapter 5. Complexity Reduction using Permutation Decoding
the correct path can be found either by the path update function or by a successful search with respect to several sequences chosen with different error locations.

**Decoding Error Probability Comparison between MCPSD and SCPSD using 8-level Quantisation**

![Graph showing decoding error probability comparison]

Figure 5.12 shows the comparison of the decoding performances between MCPSD and SCPSD.

**Complexity Comparison between MCPSD and SCPSD using 8-level quantisation**

![Graph showing complexity comparison]

Figure 5.13 shows the comparison of the decoding complexities between MCPSD and SCPSD.

---

*Chapter 5. Complexity Reduction using Permutation Decoding*
5.3 Permutation Decoding

5.3.4 Discussion

The cyclic permutation sequential decoding method provides an improvement of decoding performance without additional complexity comparing with NPSD. When a received sequence has a bad error combination in the early part of the sequence, SCPSD can easily reduce the computation by cyclic shifts. However, this method is more useful in the application to RS codes with low code rate so as to maximise the effect of the cyclic permutation. For codes with a high code rate, cyclic permutation does not influence the number of errors of the information block due to its relatively large size. In such a situation, MCPSD is more useful to overcome the shortcomings of SCPSD, providing better decoding performance and low complexity.

In MCPSD, when an overall computation limit $L$ is given, the number of trial sequences is an important parameter for achieving the most efficient performance. In the simulation results for $(7,3)$, $(15,9)$ and $(31,27)$ RS codes, the best performance has been produced by choosing the number of around half the value of the code length. In practical applications, the values will depend on the overall computational limit $L$ provided by a system. If the value of $L_c$ (computational limit per sequence) is too small due to the choice of a large $L$, the decoder may not perform sufficient computations. Thus the values of $Y$ and $L_c$ should be chosen appropriately according to a given level both of decoding performance and of complexity. Although MCPSD has an additional sorting process to assign the priorities of the cyclic permuted sequences in order of IBC, it is relatively trivial in the consideration of the achieved gains in terms of decoding performance and complexity.

As a conclusion, cyclic permutation is especially efficient for tackling certain complicated error patterns with a single long burst error in the information block at low code rate, since some error bits can be moved to the later part of the sequence. However, if there is an error pattern which is spread evenly through the sequence, cyclic permutation will be less effective, since the sequence given by a cyclic shift may have a similar error pattern to the original one. A better approach to this kind of error pattern will be described in the next chapter, resulting in further improvements of the sequential decoder.
CHAPTER 6

SQUARING AND HYBRID PERMUTATION DECODING

6.1. INTRODUCTION

We have discussed a low complexity decoding approach, for RS codes, using cyclic permutation in chapter 5. In this chapter, we introduce further permutation decoding methods and analyse their performance.

In section 6.2, another permutation technique for RS codes will be introduced, called squaring permutation. This permutation gives a different set of sequences compared with the cyclic permutation described in chapter 5. Moreover in order to preserve bit-level confidences, we employ a normal basis for symbol representation instead of the polynomial basis. Then we present squaring permutation decoding (SQPSD). By simulation, the decoding performance and complexity are demonstrated in the comparison with non-permutation decoding.

Section 6.3 is devoted to a description of a hybrid permutation technique, which is the combination of cyclic and squaring permutation. This technique provides a larger permutation group than either of the cyclic or squaring permutation individually, increasing the number of candidates which can be provided for low complexity decoding schemes. A decoding scheme is presented, called hybrid permutation sequential decoding (HPSD). Using simulation results, we analyse the performance of HPSD in terms of decoding performance and decoding complexity.

Finally, in section 6.4, there is the performance evaluation for the cyclic, squaring, and hybrid permutation decoding methods which have been presented in this thesis. The most cost-effective decoding method is chosen and its performance is evaluated compared with ML performance, as estimated by an analysis of error events.
6.2 SQUARING PERMUTATION SEQUENTIAL DECODING FOR RS CODES

Squaring permutation is a technique using the property of an idempotent (Definition 2.27) for RS codes. In this permutation, although the squaring of a code word polynomial changes the position of symbols constituting the code word, the squared result is also a code word.

Let a code word of \((n, k)\) RS codes over \(GF(2^m)\) be expressed in polynomial form as

\[
c(x) = \sum_{i=0}^{n-1} c_i x^i \quad \text{for } c_i \in GF(2^m). \tag{6.1}
\]

Using the automorphism defined in Definition 2.19, the squaring form of (6.1) is obtained as

\[
(c_0 + c_1 x + \cdots + c_{n-1} x^{n-1})^2 = (c_0^2 + c_1^2 x^2 + \cdots + c_{n-1}^2 x^{2(n-1)}). \tag{6.2}
\]

Thus an idempotent \(c(x)^2\) of \(c(x)\) can be represented by

\[
c(x)^2 = \sum_{i=0}^{n-1} c_i^2 x^{2i \mod n}. \tag{6.3}
\]

In the same way as (6.3), we can get further idempotents \(c(x)^{2^2}, c(x)^{2^3}, \ldots\). Since RS codes have \(m\) conjugates over \(GF(2^m)\), which means that the automorphism group of \(GF(2^m)\) is the cyclic group of order \(m\) (Theorem 2.6), the number of idempotents obtained becomes \(m\). That is, an idempotent of \(c(x)^{2^m}\) becomes \(c(x)\). With the previous result, this consecutive squaring produces a permutation group with \((m-1)\) permuted code words from \(c(x)\). This permutation group is called the squaring permutation group written by

\[
G_s = \bigcup_{i=0}^{m} c(x)^i. \tag{6.4}
\]

For \((n, k)\) RS codes over \(GF(2^m)\), the squaring permutation \(G_s\) from a code word \(c(x)\) is represented by

---

Chapter 6. Squaring and Hybrid Permutation Decoding
6.2 Squaring Permutation Sequential Decoding For RS codes

\[ \pi_s(\beta) \left( c(x) = \sum_{i=0}^{n-1} c_i x^i \right) = \sum_{i=0}^{n-1} c_i^{2^\beta} \cdot x^{(2^\beta i) \mod n} \quad (6.5) \]

where \( \beta = 1, 2, \ldots, m-1 \).

We consider a code word of \((7,3)\) RS codes over \(GF(2^3)\) in polynomial form as

\[ c(x) = c_0 + c_1 x + c_2 x^2 + c_3 x^3 + c_4 x^4 + c_5 x^5 + c_6 x^6, \]

where \( c_i \in GF(2^m) \). According to (6.5), the \( m \)-squaring of \( c(x) \) is obtained by

\[
\begin{align*}
\pi_s(1) \cdot c(x) &= c_0^2 + c_1^2 x + c_2^2 x^2 + c_3^2 x^3 + c_4^2 x^4 + c_5^2 x^5 + c_6^2 x^6 \\
\pi_s(2) \cdot c(x) &= c_0^2 + c_1^2 x + c_2^2 x^2 + c_3^2 x^3 + c_4^2 x^4 + c_5^2 x^5 + c_6^2 x^6 \\
\pi_s(3) \cdot c(x) &= c(x).
\end{align*}
\]

When the polynomial expression of (6.1) is represented by a vector space \( c = (c_0, c_1, \ldots, c_{n-1}) \). Table 6.1 shows the permutation of the symbol position by the squaring permutation.

**Table 6.1 Symbol Position Change by Squaring Permutation**

<table>
<thead>
<tr>
<th>( m )</th>
<th>( \beta )</th>
<th>symbol ( c_i^{2^\beta} ) at ( c = (c_0, c_1, \ldots, c_{n-1}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3</strong></td>
<td><strong>0,3</strong></td>
<td>0 1 2 3 4 5 6</td>
</tr>
<tr>
<td>1</td>
<td>0 4 1 5 2 6 3</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 2 4 6 1 3 5</td>
<td></td>
</tr>
<tr>
<td><strong>4</strong></td>
<td><strong>0,4</strong></td>
<td>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14</td>
</tr>
<tr>
<td>1</td>
<td>0 8 1 9 2 10 3 11 4 12 5 13 6 14 7</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>0 4 8 12 1 5 9 13 2 6 10 14 3 7 11</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>0 2 4 6 8 10 12 14 1 3 5 7 9 11 13</td>
<td></td>
</tr>
</tbody>
</table>

We have previously verified by an experiment as shown in section 5.2.2 that consecutive error bits in the information block need more computations to be decoded correctly than separated error bits. In the case of a burst of consecutive error bits in the information block, we have observed that cyclic permutation is helpful to deal with those error bits by moving the error burst to a later position. However, when a certain error pattern consists of widely distributed errors, cyclic permutation cannot be...
6.2 Squaring Permutation Sequential Decoding For RS codes

suitable since any cyclic shift produces a similar error pattern. At this point, the squaring permutation gives a solution for such a shortcoming of the cyclic permutation. Squaring permutation can be very effective to break such consecutive error bits or reorder the widely distributed errors so as to give a possibility that less complex error pattern can be produced.

A low complexity decoding approach[34] using squaring permutation decoding has previously been described based on algebraic decoding method. However, we need further consideration to employ this permutation for the application of our modified Fano algorithm(MFA) which is oriented on a bit-level trellis structure. We discuss details in section 6.2.1 and 6.2.2.

Normal bases for RS codes can be obtained by a method suggested by Perlis (in equation (2.6) of Definition 2.14). Table 6.2 is the comparison between polynomial and normal basis.

<table>
<thead>
<tr>
<th>Field</th>
<th>$GF(2^2)$</th>
<th>$GF(2^3)$</th>
<th>$GF(2^4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Polynomial basis</td>
<td>$\alpha^3\alpha^3\alpha^2$</td>
<td>$\alpha^3\alpha^2\alpha^2\alpha^2$</td>
<td>$\alpha^3\alpha^3\alpha^2\alpha^4$</td>
</tr>
<tr>
<td>Normal basis</td>
<td>$\alpha^3\alpha^2\alpha^3$</td>
<td>$\alpha^7\alpha^4\alpha^2\alpha^2$</td>
<td></td>
</tr>
</tbody>
</table>

On the normal basis, the representation for each element of $GF(2^n)$ is shown in Appendix B. By using the representations on normal basis, generator matrix and parity check matrix are constructed in the same way as the procedure explained in section 2.3.3.3. Table 6.3 and Table 6.4 show the symbol and binary weight distribution on normal basis for (7,3) and (7,5) RS codes. These distributions are the same as those obtained on polynomial basis as shown in Appendix C.

<table>
<thead>
<tr>
<th>Symbol Weight</th>
<th>(7,3) RS codes</th>
<th>(7,5) RS codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>245</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>1225</td>
</tr>
<tr>
<td>5</td>
<td>147</td>
<td>5586</td>
</tr>
<tr>
<td>6</td>
<td>147</td>
<td>12838</td>
</tr>
<tr>
<td>7</td>
<td>217</td>
<td>12873</td>
</tr>
</tbody>
</table>

Chapter 6. Squaring and Hybrid Permutation Decoding
Table 6.4 Binary Weight Distribution on Normal Basis for (7,3) and (7,5) RS codes

<table>
<thead>
<tr>
<th>Binary Weight</th>
<th>(7,3) RS codes</th>
<th>(7,5) RS codes</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>273</td>
</tr>
<tr>
<td>6</td>
<td>21</td>
<td>924</td>
</tr>
<tr>
<td>7</td>
<td>45</td>
<td>1956</td>
</tr>
<tr>
<td>8</td>
<td>21</td>
<td>2982</td>
</tr>
<tr>
<td>9</td>
<td>42</td>
<td>4340</td>
</tr>
<tr>
<td>10</td>
<td>126</td>
<td>5796</td>
</tr>
<tr>
<td>11</td>
<td>126</td>
<td>5796</td>
</tr>
<tr>
<td>12</td>
<td>42</td>
<td>4340</td>
</tr>
<tr>
<td>13</td>
<td>21</td>
<td>2982</td>
</tr>
<tr>
<td>14</td>
<td>45</td>
<td>1956</td>
</tr>
<tr>
<td>15</td>
<td>21</td>
<td>924</td>
</tr>
<tr>
<td>16</td>
<td>0</td>
<td>273</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>84</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>19</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>21</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

6.2.1 SQUARING PERMUTATION ON POLYNOMIAL BASIS

We examine the change of the bit contents within each symbol represented by the polynomial basis as a result of squaring permutation.

Let the $i$-th coefficient $c_i$ of equation (6.1) be an element over $GF(2^3)$ generated by primitive polynomial $p(x) = 1 + x + x^3$. The coefficient $c_i$ can be represented on polynomial basis $\gamma$ for $\gamma \in GF(2^3)$ such as

\[
\gamma_i = a_0 + a_1 \gamma + a_2 \gamma^2 \quad (6.6)
\]

where $a_j \in GF(2)$. Then the coefficients $c_i^2$, $c_i^4$, and $c_i^8$, produced by squaring permutation, can be written as

\[
c_i^2 = a_0 + a_2 \gamma + (a_1 + a_2) \gamma^2 \quad (6.7)
\]

\[
c_i^4 = a_0 + (a_1 + a_2) \gamma + a_2 \gamma^2 \quad (6.8)
\]

\[
c_i^8 = a_0 + a_1 \gamma + a_2 \gamma^2 \quad (6.9)
\]
In equations (6.7) and (6.8), we can see that squaring of a symbol results in a change of bit position or a bit addition within the original symbol. In the same way, we can get a squared symbol representation on the polynomial basis for any \((n, k)\) RS codes. If a decoding scheme is based on symbol confidence, the symbol confidence before squaring can be used for the squared symbol. However, in a decoding method employing bit confidence it is another matter, since a certain bit within a symbol is dependent on other bits. We now discuss this as follows.

Let the \(c_i = (a_0, a_1, a_2)\) in equation (6.6) have the confidences \((j_{a_0}, j_{a_1}, j_{a_2})\). The third bit in (6.7) and the second bit in (6.8) have a confidence derived jointly from adjacent bits. However, this is not desirable to obtain a bit metric for sequential decoding using bit-level confidence, since the confidence influenced by previous or later bits will give less reliable information in tracing the correct path than an independent one. Moreover, the extent to which bits are affected becomes more serious in large Galois field.

Consequently, although the symbol squaring on polynomial basis can provide a symbol permutation, it is not suitable for bit-level sequential decoding. In order to apply the squaring permutation technique to a bit-level sequential decoder, it is required that the bit confidences of the squared symbol are preserved completely.

### 6.2.2 SQUARING PERMUTATION ON NORMAL BASIS

The problem described in the previous section may be solved by using a normal basis instead of the polynomial basis. We now discuss squaring permutation on normal basis.

By definition 2.10 and 2.13, a symbol \(c_i\) over \(GF(2^3)\) can be represented on a normal basis such that

\[
c_i = a_0 \beta + a_1 \beta^2 + a_2 \beta^3 \quad \text{for } \beta \in GF(2^3)
\]

(6.10)

where \(\beta = \alpha^3\) and \(a_j \in GF(2)\). By automorphism group property in Theorem 2.6, the squaring of each term becomes

\[
(a_0 \beta)^2 = (a_0 \alpha^3)^2 = a_0 \alpha^6 = a_0 \beta^2
\]

Chapter 6. Squaring and Hybrid Permutation Decoding
Therefore the symbols of \( c_i^2 \), \( c_i^4 \), and \( c_i^8 \) obtained by squaring permutation can be written as

\[
\begin{align*}
  c_i^2 &= a_i \beta + a_0 \beta^2 + a_i \beta^2 \\
  c_i^4 &= a_i \beta^2 + a_2 \beta^2 + a_0 \beta \\
  c_i^8 &= a_0 \beta^2 + a_i \beta^2 + a_2 \beta.
\end{align*}
\]

In vector representations, each symbol is also represented as

\[
c_i(a_0, a_1, a_2) \rightarrow c_i^2(a_2, a_0, a_1) \rightarrow c_i^4(a_1, a_2, a_0) \rightarrow c_i^8(a_0, a_1, a_2). \tag{6.11}
\]

We can see from equation (6.11) that symbol squaring on the normal basis is a cyclic shift of the \( m \) bits constituting the symbol. This means that the bit confidences within the symbol are preserved by the following.

\[
c_i(j_{a_2}, j_{a_0}, j_{a_1}) \rightarrow c_i^2(j_{a_0}, j_{a_1}, j_{a_2}) \rightarrow c_i^4(j_{a_1}, j_{a_2}, j_{a_0}) \rightarrow c_i^8(j_{a_2}, j_{a_0}, j_{a_1}).
\]

This cyclic property on the normal basis can be applied in \( GF(2^m) \). This property gives a clear solution for the shortcoming by the bit addition caused on polynomial basis. Therefore the use of normal basis is suitable to implement bit-level sequential decoding using the squaring permutation. In addition, since a symbol squaring on the normal basis can be obtained by a cyclic shift, it gives a more simple procedure than on the polynomial basis. For the application of trellis sequential decoding, the systematic parity matrix based on normal basis can be used instead of polynomial basis, and there is no additional cost compared to the polynomial basis. In a practical system, the parity check matrix for constructing the trellis can be stored in ROM(read-only-memory).

### 6.2.3 SQUARING PERMUTATION SEQUENTIAL DECODER

Squaring permutation sequential decoding (SQPSD) is now introduced. The squaring permutation is carried out on the normal basis so that the bit level soft-decision can be applied directly. The MFA (modified Fano algorithm) of chapter 4 is employed. The main difference between SQPSD and MCPSD is that the \( m \) squaring-permuted
sequences are used instead of the cyclic permuted sequences in MCPSD. The decoding procedure in SQPSD is the following.

(i) Obtain \( m \) permuted sequences by the squaring permutation.

(ii) Assign the decoding priority of the \( m \) sequences in order of information block confidence (IBC) of each sequence. Set trial number to 1.

(iii) Choose as the sequence with the highest priority, the one which has the largest IBC.

(iv) Decode the chosen sequence by using MFA.

(v) Check decoding result.

- If the decoder has found a valid path satisfying the decision rule described in section 4.6.1, release the path as the decoded one and restore the permuted sequence.

- Otherwise store the best path which has been recorded so far by the path update function as described in section 4.6.1. Go to next step.

(vi) Increase trial number.

- If trial number is less than the number of possible sequences by squaring permutation, choose the sequence with next priority and then go to step (iv).

- Otherwise release the best path which has been recorded so far and then restore the permuted sequence.

In SQPSD, since the number of possible permutation sequences is \( m \) for RS codes over \( GF(2^n) \), the maximum number of trials, \( Y \), becomes \( m \). Thus for a given overall computational limit \( L \), computational limit per each sequence, \( L_c \), is

\[
L_c = \frac{L}{Y \leq m}
\]  

(6.12)

Figure 6.2 is the flow chart of SQPSD.
6.2 Squaring Permutation Sequential Decoding For RS codes

Figure 6.1 Squaring Permutation Sequential Decoder

6.2.4 SIMULATION RESULTS IN SQPSD

The simulation is carried out in the same way as described in section 5.3.3 for CPSD. It is shown how squaring permutation contributes to efficiently relocating errors and the performance of SQPSD is demonstrated through comparison with NPSD.
6.2 Squaring Permutation Sequential Decoding For RS codes

6.2.4.1 Error Location Distribution by Squaring Permutation

The error location distribution by the squaring permutation is demonstrated for (7,5) and (15,9) RS codes in the same way as described in section 5.3.3.1. Simulation procedure is the following:

- Generator and Parity check matrix for normal basis are obtained.
- Tested code words are transmitted over a Gaussian channel. The demodulator with 8-level quantizer gives a received sequence containing soft-decision values to be sent to a decoder.
- Squaring permutation generates $m$ automorphisms of the received sequence and the sequence with the largest IBC is chosen. The error location is measured in the chosen sequence.

Figure 6.2 and Figure 6.3 show that the squaring permutation in conjunction with maximum IBC relocated the errors which were evenly distributed without permutation. The number of errors within the information block has been reduced by this permutation but its scale is relatively small compared with that by cyclic permutation. This inferiority of squaring permutation can be explained by the fact that it has relatively small permutation group compared with the cyclic one. Despite such an inferiority, squaring permutation has a significance for the permutation decoding scheme because it can produce a different type of symbol permutation compared with cyclic permutation. Thus a special error pattern, which cannot be managed by cyclic permutation, may be effectively managed by squaring permutation.
6.2 Squaring Permutation Sequential Decoding For RS codes

Figure 6.2 Error Location Distribution by Squaring Permutation

Figure 6.3 Error Location Distribution by Squaring Permutation

Chapter 6. Squaring and Hybrid Permutation Decoding
6.2.4.2 Performance Analysis in SQPSD

In this section, the decoding performance and complexity are compared with NPSD (non-permutation sequential decoding) using simulation results for (7,3), (15,9), and (31,27) RS codes. The decoding parameters used in the simulation, are shown in Table 6.5 and IBC is used to decide the convenience level amongst possible candidate sequences. In addition, in order to compare the performance of SQPSD with NPSD and MCPSD in section 5.3.3.3 at the same computational limit $L$, the maximum number of trial sequences, $Y$, is chosen properly.

<table>
<thead>
<tr>
<th>Table 6.5 Decoding parameters for SQPSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>(7,3) RS codes</td>
</tr>
<tr>
<td>$\Delta T$</td>
</tr>
<tr>
<td>$L$</td>
</tr>
<tr>
<td>$Y$</td>
</tr>
<tr>
<td>$L_e$</td>
</tr>
<tr>
<td>$B$</td>
</tr>
</tbody>
</table>

Figure 6.4 and Figure 6.5 show the simulation results of SQPSD for the above three RS codes and compared with NPSD.

In Figure 6.4, it is seen that the decoding performance of SQPSD provides about 0.3 dB gain for (15,9) RS codes and about 0.8 dB gain for (31,27) RS codes at a BER of $10^{-3}$ compared with those of NPSD. In particular, for (31,27) RS codes, the decoding error rate of SQPSD is almost same as NPSD at low Eb/No but it is radically improved from Eb/No=5.0 dB. This verifies that SQPSD is more effective for RS codes over a large field and is not suitable in very noisy channels, since the number of candidate sequences for the permutation is respectively small compared with CPSD.

On the other hand, Figure 6.5 shows the complexity comparison between SQPSD and NPSD. It is observed that the computation reduction is the most effective for (15,9) RS codes. However, as shown in the comparison in Figure 6.4, more gain has been achieved with (31,27) RS codes than (15,9) RS codes at high Eb/No.
6.2 Squaring Permutation Sequential Decoding For RS codes

Decoding Error Probability Comparison between SQPSD and NPSD using 8-level Quantisation

![Graph showing decoding error probability comparison between SQPSD and NPSD](image)

Figure 6.4 Decoding Performance Comparison between SQPSD and NPSD

Complexity Comparison between SQPSD and NPSD using 8-level quantisation

![Graph showing complexity comparison between SQPSD and NPSD](image)

Figure 6.5 Decoding Complexity Comparison between SQPSD and NPSD

Chapter 6. Squaring and Hybrid Permutation Decoding
6.2.5 DISCUSSION

Squaring permutation sequential decoding (SQPSD) is a low complexity decoder to improve the decoding performance without increasing the complexity. This decoding scheme can be obtained from the normal basis representation for RS codes so that bit level soft decision values can be preserved through symbol squaring. This normal basis can be used in other cyclic block codes as well.

For the application to \((n, k)\) RS codes, since SQPSD uses \(m\) different sequences, the decoding performance will improved for long RS codes with a large Galois field. Compared with the results of MCPSD (multi-cyclic permutation decoding) as shown in Figure 5.12 and Figure 5.13, when the same overall computational limit \(L\) is given, the decoding complexity of SQPSD is almost the same as that of MCPSD, while the decoding performance is slightly worse for \((15, 9)\) and \((31, 27)\) RS codes and the same for \((7, 3)\) RS codes. This explains that SQPSD is less effective than MCPSD with the optimised decoding parameters, since the number of possible candidates sequences by squaring permutation is smaller than by cyclic permutation.

There are two interesting merits in SQPSD. One is the ability to break a long burst of consecutive errors which may cause high complexity. Moreover squaring permutation can be useful for RS codes with high code rate, for which the cyclic permutation decoder is not effective. The other merit is that SQPSD gives a solution for widespread error symbols. When some errors are spread widely in a code word, any cyclic shift does not contribute to obtaining a useful error pattern for sequential decoding. In this situation, SQPSD can solve this problem by permuting the positions of adjacent symbols. However, generally SQPSD will be worse than MCPSD due to the relatively small size of permutation group.

6.3. HYBRID PERMUTATION SEQUENTIAL DECODING

We have introduced two permutation decoding methods using cyclic and squaring permutation. In this section we introduce another permutation technique as the combination of those two permutations and a decoding method using this technique is presented.
6.3 Hybrid Permutation Sequential Decoding

6.3.1 HYBRID PERMUTATION FOR REED-SOLOMON CODES

As we have discussed in sections 5.3 and 6.2, we know for an \((n, k)\) RS code over \(GF(2^m)\) that squaring permutation provides \(m\) permuted sequences while cyclic permutation has \(n\) cyclic permuted sequences for a code word. Since each squaring permutation sequence can also generate \(n\) cyclic permuted sequences, a combination of those two permutations gives \(n \times m\) permutation sequences. This is called hybrid permutation. Table 6.6 shows the number of possible sequences by hybrid permutation. We can see that a long RS code has more available sequences because of a large field and a long code length.

<table>
<thead>
<tr>
<th>Field</th>
<th>Possible Sequences</th>
</tr>
</thead>
<tbody>
<tr>
<td>(GF(2^3))</td>
<td>21</td>
</tr>
<tr>
<td>(GF(2^4))</td>
<td>60</td>
</tr>
<tr>
<td>(GF(2^5))</td>
<td>155</td>
</tr>
<tr>
<td>(GF(2^6))</td>
<td>378</td>
</tr>
</tbody>
</table>

Hybrid permutation is very attractive to design an efficient permutation sequential decoder, since the decoder can choose the most convenient sequence from a greater variety of sequences than either of the cyclic or squaring permutation individually. This permutation gives a solution for the individual drawbacks of cyclic and squaring techniques which are used for permutation decoding. Widespread errors can be rearranged by the squaring permutation so that cyclic permutation can effectively manage the rearranged sequence. Therefore we can improve the complexity and decoding performance simultaneously. In particular, in the application of RS codes over a large Galois field, hybrid permutation will be very powerful in reducing complexity and improving decoding performance.

Figure 6.6 shows error pattern changes by hybrid-permutation for \((15,9)\) RS codes. In the figure, it is shown that four sequences, \(r(x), r(x)^2, r(x)^4\) and \(r(x)^8\), are obtained from squaring permutation and then 15 cyclic sequences are produced corresponding to each one of the squaring permutation group. Thus total 60 sequences can be obtained from a received sequence and each sequence has the same symbol and binary
weights as the original sequence because of the use of normal basis. The different thing in each sequence is the order of listed symbols. If we recall that the complexity of the sequential decoder depends on the locations of errors, it is expected that the complexity of decoding of the original sequence can also be changed with the permutation.

![Figure 6. 6 Error Pattern Changes by Hybrid Permutation](image)

**Figure 6. 6 Error Pattern Changes by Hybrid Permutation**

### 6.3.2 HYBRID PERMUTATION SEQUENTIAL DECODER

Hybrid permutation sequential decoding (HPSD) is now presented. As a trellis search algorithm, the MFA (modified Fano algorithm) of chapter 4 is again employed. For convenient sequence-first decoding based on reliability, the sequence with the greatest information block confidence (IBC) is chosen among the possible sequences in the hybrid permutation group. Moreover since the first chosen sequence cannot always have the lowest complexity, it may be efficient for the decoder to take further sequences with a smaller single trial computational limit but the same overall
6.3 Hybrid Permutation Sequential Decoding

computational limit. The way to decide the maximum number of trials $Y$ is similar to the case of SQPSD (squearing permutation sequential decoding) and MCPSD (multi-cyclic permutations sequential decoding). The decoding procedure in HPSD is the following.

(i) Obtain $m \times n$ candidates by cyclic and squaring permutation.
(ii) Assign the decoding priority of the candidates in order of IBC (information block confidence) of each candidate. Set trial number to 1.
(iii) Choose the sequence with the highest priority, which has the largest IBC.
(iv) Decode the chosen sequence by using MFA.
(v) Check decoding result.
   - If the decoder has found a valid path satisfying the decision rule, release the path and restore its sequence order.
   - Otherwise store the best path which has been recorded so far by the path update function. Then go to the next step.
(vi) Increase trial number.
   - If trial number is less than a given maximum-trial-number, choose the sequence with next priority and then go to step (iv).
   - Otherwise release the best path which has been recorded so far and then restore the permuted sequence with respect to the path.

The hybrid permutation sequential decoder will deal with an error pattern in a received sequence most efficiently by using cyclic or squaring permutation as appropriate. This decoding method will be useful for low complexity sequential decoding.
6.3 Hybrid Permutation Sequential Decoding

6.3.3 SIMULATION RESULTS IN HPSD

The simulation is carried out in the same way as the previous ones in CPSD and SQPSD. It is also shown the relocation of errors by hybrid permutation and the performance of HPSD is demonstrated through comparison with NPSD.

---

Chapter 6. Squaring and Hybrid Permutation Decoding
6.3.3.1 Error Location Distribution by Hybrid Permutation

The error location distribution by hybrid permutation is demonstrated for (7,5) and (15,9) RS codes. Simulation procedure is the same as for squaring permutation in 6.2.4.1 except that hybrid permutation is used.

Figure 6.8 and Figure 6.9 show that the hybrid permutation relocates the errors which are evenly distributed at each bit position in non-permutation. It is shown that error occurrences within the information block have been much lower than within the parity check block when using hybrid permutation. Moreover, the occurrence of errors at the first symbol is relatively low. Thus, the sequence chosen by hybrid permutation can be used as the most convenient sequence for sequential decoding.

Comparing with the results by cyclic permutation in section 5.3.3.1 and squaring permutation in section 6.2.4.1, the relocation by the hybrid permutation is the most effective by further lowering the incidence of error events in the middle of the information block.

![Error Relocation by Hybrid Permutation for (7,5) RS codes at 4.0 dB with Q=8](image)

Figure 6.8 Error Location by Hybrid Permutation
Error Relocation by Hybrid Permutation for (15,9) RS codes at 4.0 dB with Q=8

Figure 6.9 Error Location by Hybrid Permutation

6.3.3.2 Performance Analysis in HPSD

In this section, the simulation results of HPSD are obtained for some RS codes and their performance is analysed in terms of the decoding performance and complexity in the same way as the cases of CPSD in section 5.3.3 and SQPSD in section 6.2.4.2. In the simulation, the decoding parameters used are shown in Table 6.7.

Table 6.7 Decoding Parameters for HPSD

<table>
<thead>
<tr>
<th>RS codes</th>
<th>$\Delta T$</th>
<th>$B$</th>
<th>$L$</th>
<th>Trellis State Number</th>
<th>Code word Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>(7,3)</td>
<td>4.0</td>
<td>2.0</td>
<td>$2^8$</td>
<td>$2^9$</td>
<td>$2^9$</td>
</tr>
<tr>
<td>(7,5)</td>
<td>4.0</td>
<td>0.0</td>
<td>$2^7$</td>
<td>$2^6$</td>
<td>$2^{15}$</td>
</tr>
<tr>
<td>(15,9)</td>
<td>3.0</td>
<td>7.0</td>
<td>$2^{18}$</td>
<td>$2^{24}$</td>
<td>$2^{36}$</td>
</tr>
<tr>
<td>(15,11)</td>
<td>3.0</td>
<td>3.0</td>
<td>$2^{16}$</td>
<td>$2^{16}$</td>
<td>$2^{24}$</td>
</tr>
<tr>
<td>(15,13)</td>
<td>3.0</td>
<td>1.0</td>
<td>$2^{14}$</td>
<td>$2^8$</td>
<td>$2^{22}$</td>
</tr>
<tr>
<td>(31,23)</td>
<td>5.0</td>
<td>10.0</td>
<td>$2^{25}$</td>
<td>$2^{40}$</td>
<td>$2^{115}$</td>
</tr>
<tr>
<td>(31,25)</td>
<td>5.0</td>
<td>8.0</td>
<td>$2^{24}$</td>
<td>$2^{30}$</td>
<td>$2^{135}$</td>
</tr>
<tr>
<td>(31,27)</td>
<td>5.0</td>
<td>6.0</td>
<td>$2^{21}$</td>
<td>$2^{20}$</td>
<td>$2^{135}$</td>
</tr>
<tr>
<td>(31,29)</td>
<td>5.0</td>
<td>4.0</td>
<td>$2^{19}$</td>
<td>$2^{10}$</td>
<td>$2^{145}$</td>
</tr>
</tbody>
</table>
6.3.3.2.1 Decoding Performance according to the Maximum Number of Trial Sequences $Y$

The performance of HPSD is optimised by properly choosing more than one sequence amongst many candidates provided by hybrid permutation. Thus we need to find the optimised values of the maximum number of trial sequences, $Y$, for (15,9) and (31,27) RS codes. The simulation is carried out with different values of $Y$ under the same computational limit $L$ for a given code. Decoding error rate, $P_e$, and average path extensions per information bit, $\overline{C}$, are measured as a function of $Eb/No$.

Table 6.8 and Table 6.9 are the simulation results for (15,9) and (31,27) RS codes at different values of $Eb/No$. In Table 6.8, we can see that the decoding error rate $P_e$, for (15,9) RS codes, gradually decreases and the complexity also grows with the growth of $Y$. In order to decide the most efficient combinations, we need to inspect the amount of the decoding performance improvement at the cost of the additional complexity. Thus the choice of the combination of $Y=4$ and $L_e=2^{16}$ will be appropriate for these codes. In the same way, we choose the optimal values for (31,27) RS codes in Table 6.9. In this case, we can choose the $Y=16$ as the best one for (31,27) RS codes.

In the application of HPSD, the use of the optimised sequences will contribute to the most cost-effective performance. In a practical system which may not allow computations, the decoding performance can be maximised by the appropriate combination of the parameters $Y$ and $L_e$.

Table 6.8 Decoding Results for (15,9) RS codes according to different $Y$

<table>
<thead>
<tr>
<th>$Y$</th>
<th>$L_e$</th>
<th>$P_e$</th>
<th>$C$</th>
<th>$P_e$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$2^{18}$</td>
<td>0.004</td>
<td>733.3</td>
<td>0.0004</td>
<td>170.3</td>
</tr>
<tr>
<td>2</td>
<td>$2^{17}$</td>
<td>0.004</td>
<td>808.8</td>
<td>0.00036</td>
<td>203.6</td>
</tr>
<tr>
<td>4*</td>
<td>$2^{16}$</td>
<td>0.0029</td>
<td>977.6</td>
<td>0.00019</td>
<td>246.4</td>
</tr>
<tr>
<td>8</td>
<td>$2^{15}$</td>
<td>0.0027</td>
<td>1143.5</td>
<td>0.00018</td>
<td>306.8</td>
</tr>
</tbody>
</table>

* indicates optimised parameter.
Table 6.9 Decoding Results for (31,27) RS codes according to different \( Y \)

<table>
<thead>
<tr>
<th>( Y )</th>
<th>( L_e )</th>
<th>( \frac{E_b}{N_0} = 4.0 \text{ dB} )</th>
<th></th>
<th>( \frac{E_b}{N_0} = 5.0 \text{ dB} )</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( 2^{21} )</td>
<td>0.0050</td>
<td>6484.5</td>
<td>0.00088</td>
<td>450.5</td>
</tr>
<tr>
<td>2</td>
<td>( 2^{20} )</td>
<td>0.0042</td>
<td>2997.2</td>
<td>0.00034</td>
<td>456.1</td>
</tr>
<tr>
<td>4</td>
<td>( 2^{19} )</td>
<td>0.0034</td>
<td>3538.5</td>
<td>0.00013</td>
<td>487.3</td>
</tr>
<tr>
<td>8</td>
<td>( 2^{18} )</td>
<td>0.0034</td>
<td>4047.6</td>
<td>0.00086</td>
<td>574.2</td>
</tr>
<tr>
<td>16*</td>
<td>( 2^{17} )</td>
<td>0.0003</td>
<td>4299.1</td>
<td>0.00074</td>
<td>645.5</td>
</tr>
<tr>
<td>32</td>
<td>( 2^{16} )</td>
<td>0.0026</td>
<td>4494.3</td>
<td>0.00081</td>
<td>710.1</td>
</tr>
</tbody>
</table>

* indicates optimised parameter.

6.3.3.2.2 Cost-effective Performance in HPSD

In this section, the performance of HPSD is shown and is compared with that of NPSD for (7,3), (15,9) and (31,27) RS codes.

Figure 6.10 is the decoding performance comparison between HPSD and NPSD for (7,3), (15,9) and (31,27) RS codes. We see that HPSD provides a BER of \( 10^{-5} \), about extra 0.6 dB gain for (15,9) RS codes, about 1.3 dB extra gain for (31,27) RS codes, and almost same performance for (7,3) RS codes as NPSD.

Figure 6.11 is the complexity comparison between HPSD and NPSD for (7,3), (15,9) and (31,27) RS codes. We can see that HPSD gives very low complexity compared with NPSD.

In Figure 6.12, the complexity of HPSD is compared with that of MCPSD for some length 31 RS codes with the same computational limit \( L \). We can see that HPSD produces much less complexity for (31,23) RS codes than MCPSD. This is explained by the fact that in the case that a computational limit cannot be sufficient because of high decoding complexity, the HPSD can produce the most cost-effective performance. In this case the high complexity is caused by the large number of trellis states, and MCPSD using cyclic permutation is not enough to pick up the best candidate.
Table 6.10 is the decoding error rate for (31,23) and (31,25) RS codes. It is shown that HPSD provides much lower decoding error rate than MCPSD. In this result, the decoding error rate for (31,23) RS codes is worse than (31,25) RS codes. This results from the fact that we used for (31,23) RS codes, a computational limit \( L \) two times as large as for (31,25) RS codes but the number of trellis states of the (31,23) RS codes is 2\(^6\) times as large as for (31,25) RS codes. Thus sufficient computations could not performed and most of the error source occurred due to decoding overflow by computation limit. If sufficient computations are allowed for (31,23) RS codes, the decoding performance will be improved, with an increase in complexity.

Table 6.10 Decoding Error Rate for (31,23) and (31,25) RS codes

<table>
<thead>
<tr>
<th></th>
<th>(31,23) RS codes</th>
<th>(31,25) RS codes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MCPSD</td>
<td>HPSD</td>
</tr>
<tr>
<td>( \frac{E_b}{N_0} = 4.0 \text{ dB} )</td>
<td>0.0026</td>
<td>0.00095</td>
</tr>
<tr>
<td>( \frac{E_b}{N_0} = 5.0 \text{ dB} )</td>
<td>0.00017</td>
<td>0.00002</td>
</tr>
</tbody>
</table>

Decoding Error Probability Comparison between HPSD and NPSD using 8-level Quantisation

![Decoding Performance Comparison between HPSD and NPSD](image)

Figure 6.10 Decoding Performance Comparison between HPSD and NPSD
6.3 Hybrid Permutation Sequential Decoding

Figure 6.11 Decoding Complexity Comparison between HPSD and NPSD

Figure 6.12 Complexity Comparison for length 31 RS codes
6.3.4 DISCUSSION

Hybrid permutation decoding shows great improvement in decoding performance and complexity by using the properties of cyclic and squaring decoding with a large number of candidates. Any type of error pattern can be managed effectively in this scheme. Moreover HPSD can maximise decoding performance in an application when sufficient computations are not allowed due to limited system resources or decoding time.

In the inspection of the simulation results in the previous section, a great complexity advantage of HPSD has been shown for (31,25) and (31,23) RS codes, since those codes need an enormous number of computations due to a large number of trellis states. In such a situation, HPSD using a large number of permutation sequences is very efficient for the decoder to find the correct path instead of with a few sequences. Moreover when the decoder faces a deadlock in which it cannot perform properly because of repeated threshold adjustment owing to severe noise, HPSD leads to the desirable effect of escaping such a situation without wasting computations by giving up the sequence with high complexity at an early stage and trying other candidate sequence.

For a given computational limit, the choice of $Y$ is important to achieve the best result. If the number of computations per sequence is too small because of large values of $Y$, this may increase complexity without performance improvement. On the other hand, if too small value of $Y$ is chosen, this would not use other candidates in the hybrid permutation group. In general, for the decoding of RS codes with low code rate, large $Y$ is more effective than for high code rate, since the low rate codes have a larger number of trellis states which increases the decoding complexity.

6.4. PERFORMANCE EVALUATION

In this section, decoding performance and complexity is evaluated for the three permutation decoding methods, cyclic, squaring, and hybrid permutation. For decoding performance evaluation, an estimation of ML performance is obtained by the analysis of error events which has been presented from the simulation. On the other
6.4 Performance Evaluation

In the decoding simulation, any decoding error is detected by the binary addition $\oplus$ of a transmitted code word $t(x)$ and a decoded code word $d(x)$.

$$e(e_0, e_1, \cdots, e_{n-1}) = t(t_0, t_1, \cdots, t_{n-1}) \oplus d(d_0, d_1, \cdots, d_{n-1})$$  \hspace{1cm} (6.13)

where $e, t, d \in GF(2)$. The number of 1s in vector $e$ indicates the number of errors.

When a received sequence is decoded incorrectly, the error event is one of three possible error event types, uncorrectable, wrong path search, and decoding overflow as discussed in section 4.5.3. Thus if we calculate the BER according to each event, the decoding of the sequential decoder can be evaluated.

Let $M_t$ be the path metric of the transmitted code word and $M_d$ denote the path metric of a decoded code word. If $M_d \geq M_t$, this error source is an uncorrectable error event which occurs even in ML decoding. This error event is represented by $P_e$.

On the other hand, if $M_d < M_t$, the decoder overflow event is checked, which means that the decoding has been stopped by a fixed computational limit. This type of error can be represented by the probability $P_c$. Otherwise the error event by wrong path search is declared with the probability $P_w$, since the decoder has chosen a wrong path even though a sufficient computations were allowed. Figure 6.13 is the flow chart for the error event classification according to the decoding result of each received sequence, using $M_t$ and $M_d$.
$M_t$: the path metric of the transmitted code word.

$M_d$: the path metric of the decoded code word.

Figure 6.13 Error Event Classification

With a ML decoder, there would be only the uncorrectable event. Let $P_e$ be overall decoding error probability. The difference between $P_e$ and $P_v$ indicates how far the performance of a sequential decoder is away from ML decoder. Thus if the difference approaches zero by minimising the probability $P_e$ and $P_w$ of other two error events, we can obtained near-ML performance as

$$P_e = P_v \quad (6.14)$$

where $(P_e + P_w) \ll P_v$.

In order to justify this evaluation scheme, we compare the decoding performance between HPSD and Viterbi decoding which is implemented on Wolf’s trellis. In Table 6.11, we can see that the decoding performance is almost same as that of the Viterbi algorithm for (7,3) RS codes. This is explained by the fact that the probability $P_e$ in HPSD dominates the overall decoding probability $P_v$. Thus it is verified that when the decoding performance of a sequential decoder has the relationship of equation (6.14), this decoder provides almost ML performance.
Table 6.11 Comparison between HPSD and Viterbi Decoder for (7,3) RS codes

<table>
<thead>
<tr>
<th>Eb/No (dB)</th>
<th>( P_c )</th>
<th>( P_e )</th>
<th>( P_{e_{av}} )</th>
<th>( C )</th>
<th>( P_c )</th>
<th>( P_e )</th>
<th>( C )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.0</td>
<td>0.075</td>
<td>0.002</td>
<td>0.009</td>
<td>13.0</td>
<td>0.087</td>
<td>56.8</td>
<td></td>
</tr>
<tr>
<td>4.0</td>
<td>0.0017</td>
<td>0.000006</td>
<td>0.0043</td>
<td>7.6</td>
<td>0.002</td>
<td>56.8</td>
<td></td>
</tr>
<tr>
<td>5.0</td>
<td>0.0029</td>
<td>0.000006</td>
<td>0.00002</td>
<td>3.7</td>
<td>0.0003</td>
<td>56.8</td>
<td></td>
</tr>
</tbody>
</table>

Next we evaluate the simulation results for different RS codes according to the comparison between uncorrectable error event and overall error event. Since HPSD provides the best performance and lowest complexity compared with other methods, MCPSD, SCPSD, or SQPSD, we analyse the decoding performance obtained from HPSD.

In Figure 6.14, we can see that the gap between the probabilities of \( P_c \) and \( P_e \) is almost zero for (7,3) RS codes and there is a slight difference for (7,5) RS codes. It is said that HPSD produces near-ML performance for (7,3) and (7,5) RS codes.

In Figure 6.15, it is clearly seen that ML performance has been achieved for (15,13) and (15,11) RS codes because there is no difference between \( P_c \) and \( P_e \). For (15,9) RS codes, the slight gap of around 0.1 dB, indicates a boundary of an ideal ML performance. In fact, the error event by wrong path search or by computational limit can be a part of the error event from a ML decoder. If the gap between \( P_c \) and \( P_e \) is trivial, it can be said that HPSD produces near-ML performance.

Figure 6.16 shows the results for (31,23), (31,25), (31,27), and (31,29) RS codes. In the same way as the previous analysis, HPSD provides ML performance for (31,27) and (31,29) RS codes. On the other hand, for (31,23) and (31,25) RS codes in which code rates are relatively low, the gap between \( P_c \) and \( P_e \) is so wide that their ML performance cannot be estimated. Moreover the performance for (31,23) RS codes is worse than for (31,27) RS codes despite better error correcting capability. This results from the fact, that since (31,23) RS codes have \( 2^{19} \) times more trellis states as (31,25) RS codes, sufficient computations could not carried out since we used for (31,23) RS codes twice the computational limit of (31,25) RS codes.
Figure 6.14 Performance Evaluation for (7,3) and (7,5) RS codes

Figure 6.15 Performance Evaluation for (15,19), (15,11) and (15,13) RS codes
6.4 Performance Evaluation

ML performance Estimation by Error Events using 8-level Quantisation

![Graph showing performance evaluation for various RS codes](image)

Figure 6.16 Performance Evaluation for (31,23),(31,25),(31,27) and (31,29) RS codes

6.4.2 COMPLEXITY ANALYSIS

6.4.2.1 Computational Distribution according to Permutation Methods

The computational distributions for (7,3) and (15,9) RS codes are measured according to sequential decoding with different permutation techniques. In order to see the effect of the different permutations, we used the standard Fano algorithm without the forced backtracking behaviour caused by the decision rule in section 4.5.2. The decoder chooses only one sequence with the largest IBC amongst the permutation group for the decoding. When the decoder has found a path, the number of computations required, $C$, is measured and the event is counted when $C > X$ where $X$ is an appropriately fixed number.

From Figure 6.17 and Figure 6.18, we can see that the sequential decoder with hybrid permutation needs the lowest computations; the second lowest is the decoder using cyclic permutation and the third lowest is squaring permutation. This result verifies that IBC is a useful criterion to pick up the sequence which is predicted as the lowest complexity amongst permutation group, and the number of possible candidates of a
6.4 Performance Evaluation

permutation group provides more possibility that the sequence with the lowest complexity can be found.

Figure 6.17 Comparison of computational distribution for (7,3) RS codes

Figure 6.18 Comparison of computational distribution for (15,9) RS codes.

Chapter 6. Squaring and Hybrid Permutation Decoding
6.4.2.2 Complexity Comparison of Permutation Decoding Methods

For four permutation sequential decoding methods, SCPSD, MCPSD, SQPSD, and HPSD, the complexities are compared for some RS codes. The complexity is represented by average computations per an information bit $C$. Simulation is carried out with the same decoding parameters of $\Delta T$, $B$, and $L$ (overall computational limit) for (7,3), (15,9), and (31,27) RS codes.

In Figure 6.19, we can see that HPSD provides the lowest complexity among those permutation decoding methods. In the comparison with MCPSD, the complexity of HPSD is slightly better. However, the advantage of HPSD will be the further improvement of decoding performance even at lower complexity. In addition, in the case when the decoder cannot have a sufficient computational limit, HPSD contributes to improving decoding performance and reducing computations.

![Complexity Comparison by different permutations using 8=level quantisation](image)

Figure 6.19 Complexity comparison according to different permutation decoding methods

---

Chapter 6. Squaring and Hybrid Permutation Decoding
6.4.3 DISCUSSION

The analysis by error events is a simple method to assess the decoding performance compared with the ML decoder. When the overall error event probability is dominated by uncorrectable error events in which a decoding path is more likely than the transmitted code word, this result produces the ML performance. This analysis can be a useful measure to investigate the decoding performance of a decoder. Furthermore, there is the lucky case in which the sequential decoder can find the correct path which must be rejected if the decoder search fully for possible paths, as Viterbi decoding does. Thus the bit error rate by uncorrectable errors event may even be less than that of the ML decoder. This means that the sequential decoder may perform better than the ML decoder only when the overall error event is dominated by uncorrectable errors.
CHAPTER 7

CONCLUDING REMARKS AND FURTHER WORK

7.1. INTRODUCTION

Complexity and decoding performance are important factors in evaluating a decoding method. A desirable decoder property is that coding gain is as great as possible and complexity as low as possible. Since maximum coding gain can be achieved by employing soft-decision information, we are always interested in soft-decision decoding with low complexity.

In this thesis, we have presented some bit-level soft-decision decoding methods for low complexity and near-ML performance. In these methods, a modified Fano sequential decoding algorithm has been introduced and three sequential decoders employing cyclic, squaring, and hybrid permutations, have been implemented to reduce the complexity of the sequential decoding.

These new decoding methods can fully use the bit-level soft-decision information and contribute to impressive complexity reduction due to the reliability-first search scheme using those permutation techniques. In the following section, we give a conclusion on research goals which have been achieved in this thesis.

7.2. CONCLUSION

The trellis structure used for decoding has binary branches, which is beneficial in the following two ways. One is that on the binary-branch it is very easy to employ full bit-level soft-decision information. The other is that a binary branch at each node reduces the computations performed by the back tracking operation of the Fano algorithm. Although the width of the trellis is greater, since the complexity of the Fano algorithm is independent of the width and depends on the number of trellis states, this trellis structure is very suitable for the design of a low complexity sequential decoder.
The modifications of path update and decision rule functions, which are used for the improvement of decoding performance, allow the achievement of near-ML performance. The path update function always releases the most likely path as long as the decoder finds it within a fixed computational limit. It is therefore crucial that the decoder scans a likely region which includes the correct path. On the other hand, the decision rule minimises the possibility that the decoder has identified a wrong path. When a path has been found with a poor path metric, the decoder initiates forced backtracking operations to find a path more likely than the path chosen. With those two modifications, if we assume that the correct path has been tried by the decoder, the decoding result will be the same performance as in an ML decoder. However, in the case when the decoder has not searched for the correct path because of an insufficient computational limit, there is a decoding performance degradation. Thus it is essential in the improvement of performance and complexity for the decoder to focus on a region including the correct path.

Permutation groups provide sequences which are equivalent to the received one. Permutation decoding is a decoding scheme based on the convenient sequence-first search principle. IBC (information block confidence) is a measure to represent the convenience level of a sequence. Since a sequence with larger IBC will contain a statistically more reliable information block than others, this measure can be useful to pick up a sequence amongst a particular permutation group, even though, in the RS codes with high code rate, the convenience estimation by IBC will be less effective.

Cyclic permutation decoding is very efficient at changing an error pattern of the received sequence into other patterns which may give lower complexity. In the application to RS codes, this permutation decoding method reduces the complexity relative to non-permutation decoding by about 33% in length 7 RS codes, whilst producing the same ML performance as NPSD. In length 15 and 31 RS codes, the complexity was also lowered even with the considerable improvement in decoding performance. The effect of cyclic permutation depends on which of two cases applies. When the given computational limit is insufficient, cyclic permutation improves the decoding performance by trying more likely paths without complexity growth. On the other hand, when the computational limit is enough to search for the correct path,
cyclic permutation decoding reduces the computations required to follow the correct path, since the decoder can find the correct path at an early stage of trellis searching. However, the convenience estimation using IBC cannot always give the best sequence and it is not effective for RS codes with high code rate. Therefore MCPSD using several sequences is more efficient in the aspects of decoding performance and complexity than SCPSD using a single sequence.

Squaring permutation decoding has also shown low complexity without performance loss compared with non-permutation decoding. One noticeable point in the squaring permutation is the use of the normal basis for the representation of RS codes. This basis preserves bit-level soft-decision information so that the result of squaring a symbol becomes a cyclic shift of bit constituents of that symbol. This normal basis will be applicable to any non-binary cyclic codes. In general the squaring permutation decoding method has shown worse complexity and decoding performance compared with cyclic permutation decoding, since it has a smaller permutation group. Despite this shortcoming, this decoding method can be useful for decoding certain received sequences with wide spread error bits, which cyclic permutation decoding cannot deal with.

Hybrid permutation decoding gives the best solution for low complexity sequential decoding. Since this method includes the merits of both cyclic and squaring permutation decoding, it produced the best decoding performance and the lowest complexity. In the simulation results, this decoding algorithm showed near-ML performance for (7,3), (7,5), (15,9), (15,11), (15,13), (31,27), and (31,29) RS codes. For (31,23) and (31,25) RS codes, the decoding performance did not approach ML performance. The reason is that sufficient computations for those codes cannot be allowed in the simulation. If the computational limit used increases, the decoding performance will be improved and approach the ML performance. The great advantage of HPSD is that there are a large number of permutation groups for hybrid permutation and thus it is very effective at overcoming the individual drawbacks of cyclic and squaring permutation decoding. Therefore HPSD can provide the lowest complexity sequential decoder for any RS codes. In particular, this decoding method
7.2 Conclusion

is powerful for long RS codes which must normally expend an enormous number of computations due to the large number of trellis states.

Additional complexity is incurred in finding the most convenient sequence. The three permutation decoding methods have a sorting procedure to assign the convenience level for candidate sequences of a permutation group. However, this complexity cost is trivial in the overall decoding procedure. The reason is that a sequential decoder may suffer from repeated searching during a certain section which consists of bad error pattern and the number of computations for passing through the serious error block would be very large due to the adjustment of decoding parameters. In general, average decoding complexity is influenced by decoding complexity at the worst case of error pattern and this additional cost can be allowed.

As a conclusion, HPSD is a very cost-effective decoding method approaching ML performance with low complexity, and has the following advantages:

- Availability of bit-level soft-decision with a simple binary-trellis structure
- Near-ML performance with reasonable computation for many long RS codes
- Low complexity decoding by reliability-first search using hybrid permutation techniques
- Adaptability according to channel signal to noise ratio

These results can be applied to any non-binary cyclic block codes. In addition, the complexity and decoding error rate can be improved by the use of an optimised quantizer, and finer criteria for convenient sequence search. We note that the choice of the optimised parameters of maximum trial sequence $Y$ and computational limit $L$ is very important for giving the most cost-effective performance.

7.3. FURTHER WORK

Further work for practical application of HPSD is suggested as follows. Firstly HPSD is applied to shortened RS codes in which trellis state complexity is lowered. This approach helps HPSD to achieve near-ML performance for long RS codes with a large number of states, at the expense of code rate. Secondly HPSD is employed for a
scheme using multiple coding. As shown in this thesis, since HPD produces ML-performance for many RS codes except high complexity ones such as (31,23) and (31,25) RS codes, this approach will be effective.

7.3.1 SHORTENED REED-SOLOMON CODES USING HPD

We have concluded that hybrid permutation decoding gives near-ML performance with low complexity for many long RS codes. However, for RS codes with low code rate such as (31,23) and (31,25), the computations required are still enormous to the extent that the simulation cannot be supported. We therefore need another approach to solve the problem. In such a situation, we can introduce a method to reduce the searching region of the trellis. This method is called shortening of RS codes.

An \((n, k)\) RS code can be shortened by filling zero symbols in \(u\) information symbols in the encoding procedure, from which we can obtain an \((n-u, k-u)\) shortened RS code. At the transmission, only \(k-u\) information symbols (\(u\) zero symbols are deleted) are sent to a receiver and then the deleted zero symbols are refilled before decoding. At the decoder, this sequence is regarded as a received sequence of length \(n\) which has zero symbols at certain coordinates. Thus HPD can be directly applied to the sequence.

We observe that the minimum distance of this shortened RS code is at least as great as that of the unshortened RS code. The number of possible trellis states is reduced depending on the amount shortened, \(u\). For a \((29,21)\) shortened RS code obtained from the \((31,23)\) RS code, the possible number of code words which are searched by HPD can be reduced considerably even through the minimum distance of the shortened code is the same as the \((31,23)\) RS code.

Finally, we note that there are many possibilities that hybrid permutation techniques can be used effectively with shortened RS codes according to the coordinates of zero symbols filled. Since the number of errors in the information block is the most critical factor, it will be a reasonable method to take the zero symbols at the start of the sequence as shown in Figure 7.1. In the decoding procedure, the refilled zero symbols are indicated on a sequence obtained from cyclic or squaring permutations. There is
only one path corresponding to those zero symbols. Thus when the decoder meets the zero symbols, it just moves forward or backward over the trellis levels for these symbols without the calculation of the path metric.

**Encoder:**

| 1 zero symbols | (k-l) information symbols | (n-k) parity check symbols |

**Transmitter:**

| (k-l) information symbols | (n-k) parity check symbols |

**Demodulator:**

| (k-l) information symbols | (n-k) parity check symbols |

**Decoder:**

| 1 zero symbols | (k-l) information symbols | (n-k) parity check symbols |

Figure 7.1 (n-l, k-l) shortened RS codes

### 7.3.2 Subcode of Reed-Solomon Codes Using HPSD

A class of subcode for RS codes can be constructed by inserting one zero bit in each symbol of the information block. In this scheme, the code rate is

\[ R = \frac{k(m-1)}{mn-k} \]

Since a subcode satisfies all characteristics of the original RS codes, HPSD can be used without any additional procedure. In the decoding procedure, trellis searching is performed on the subspace \( S \) and thus decoding complexity can be reduced considerably. On the trellis, a node in which a zero bit is inserted has only one path toward zero and thus there is no alternative searching at the node. This is of considerable advantage in sequential decoding operations.

The additional bit can be used in other ways instead of being set to zero. One is that we can use the bit to indicate the parity of each symbol. In this scheme, the decoder can detect whether a symbol searched is wrong and take further action at that stage. For example, the decoder can initiate a back tracking operation without further

*Chapter 7. Concluding Remarks and Further Work*
searching until the correct symbol has been found. This can be helpful to reduce the high complexity caused by error occurrence at an early state of a given sequence. In addition this scheme prevents the decoder digressing from the correct path for a long time. Although this scheme is not useful if the parity bit is corrupted, a parity check error gives a clear indication that the current symbol is likely to be wrong. Further consideration and evaluation of this idea is required, however.

7.3.3 CONCATENATED CODING SCHEME USING HPSD

Concatenated coding is a technique, introduced by Forney[15], employing multiple levels of coding for implementation of a code with a very long block length and a large error-correcting capability. An advantage of such a scheme is a significant reduction in complexity needed to provide the same overall error rate.

![Concatenated Coding Scheme](image)

Figure 7.2 Concatenated Coding Scheme

Figure 7.2 illustrates the data procedure of a concatenated coding scheme. In the figure, the outer code is an \((n, k)\) non-binary code consisting of \(K\)-bit symbols and an \((N, K)\) block code is used as the inner code. The concatenated code has the following parameters:

<table>
<thead>
<tr>
<th>Overall Code length</th>
<th>(nN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Bits</td>
<td>(kK)</td>
</tr>
<tr>
<td>Code Rate</td>
<td>(\frac{kK}{nN})</td>
</tr>
</tbody>
</table>

RS codes are most commonly used as the outer code associated with many different inner codes, since these codes provide maximum value of minimum distance. Thus an
efficient decoding method for RS codes will influence the overall decoding performance of the scheme in terms of error-correcting performance and complexity. The main roles for the outer code and inner code are burst error correction and random error correction respectively. On this point, HPSD is very useful for the correction of a burst error or a random error by using hybrid permutation. In fact, HPSD is better for random errors compared with symbol level decoding, since the sequential decoding is carried out at bit-level. Thus HPSD will be used not only for the outer code but also for an inner block code to which HPSD can be applied. More reliable output obtained from the inner decoder will contribute to a more powerful performance of HPSD than in single level decoding. Finally, methods of soft output decodings in HPSD could be developed for use in iterative decodings of concatenated codes.
APPENDIX A

GENERATOR POLYNOMIAL FOR RS CODES

\[ G(X)_{(7,3)} = \prod_{i=1}^{\lfloor \frac{7}{2} \rfloor} (X + \alpha^i) = X^4 + \alpha^3 X^3 + X^2 + \alpha X + \alpha^3 \]

\[ G(X)_{(7,5)} = \prod_{i=1}^{\lfloor \frac{7}{2} \rfloor} (X + \alpha^i) = X^2 + \alpha^4 X + \alpha^3 \]

\[ G(X)_{(15,9)} = \prod_{i=1}^{\lfloor \frac{15}{2} \rfloor} (X + \alpha^i) = X^6 + \alpha^{10} X^5 + \alpha^{14} X^4 + \alpha^4 X^3 + \alpha^6 X^2 + \alpha^9 X + \alpha^6 \]

\[ G(X)_{(15,11)} = \prod_{i=1}^{\lfloor \frac{15}{2} \rfloor} (X + \alpha^i) = X^4 + \alpha^{13} X^3 + \alpha^6 X^2 + \alpha^3 X + \alpha^{10} \]

\[ G(X)_{(15,13)} = \prod_{i=1}^{\lfloor \frac{15}{2} \rfloor} (X + \alpha^i) = X^2 + \alpha^5 X + \alpha^3 \]

\[ G(X)_{(31,23)} = \prod_{i=1}^{\lfloor \frac{31}{2} \rfloor} (X + \alpha^i) = X^8 + \alpha^3 X^7 + \alpha^{23} X^6 + \alpha^{23} X^5 + \alpha^{19} X^4 + \alpha X^3 + \alpha^9 X^2 + \alpha^{30} X + \alpha^5 \]

\[ G(X)_{(31,25)} = \prod_{i=1}^{\lfloor \frac{31}{2} \rfloor} (X + \alpha^i) = X^6 + \alpha^{10} X^5 + \alpha^9 X^4 + \alpha^{24} X^3 + \alpha^{16} X^2 + \alpha^{24} X + \alpha^{21} \]

\[ G(X)_{(31,27)} = \prod_{i=1}^{\lfloor \frac{31}{2} \rfloor} (X + \alpha^i) = X^4 + \alpha^{24} X^3 + \alpha^{19} X^2 + \alpha^{29} X + \alpha^{10} \]

\[ G(X)_{(31,29)} = \prod_{i=1}^{\lfloor \frac{31}{2} \rfloor} (X + \alpha^i) = X^2 + \alpha^{19} X + \alpha^3 \]
### APPENDIX B

**GALOIS FIELD REPRESENTATION FOR RS CODES**

**B.1** $GF(2^3)$ using $p(x) = x^3 + x + 1$ and $\alpha$ and $\beta \in GF(2^3)$

<table>
<thead>
<tr>
<th>Power of $\beta$</th>
<th>Polynomial Basis ($\alpha^2 \alpha^1 \alpha^0$)</th>
<th>Normal Basis ($\alpha^5 \alpha^6 \alpha^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>001</td>
<td>111</td>
</tr>
<tr>
<td>1</td>
<td>010</td>
<td>110</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
<td>101</td>
</tr>
<tr>
<td>3</td>
<td>011</td>
<td>001</td>
</tr>
<tr>
<td>4</td>
<td>110</td>
<td>011</td>
</tr>
<tr>
<td>5</td>
<td>111</td>
<td>100</td>
</tr>
<tr>
<td>6</td>
<td>101</td>
<td>010</td>
</tr>
</tbody>
</table>

**B.2** $GF(2^4)$ using $p(x) = x^4 + x + 1$ and $\alpha$ and $\beta \in GF(2^4)$

<table>
<thead>
<tr>
<th>Power of $\beta$</th>
<th>Polynomial Basis ($\alpha^3 \alpha^2 \alpha^1 \alpha^0$)</th>
<th>Normal Basis ($\alpha^5 \alpha^6 \alpha^3$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0001</td>
<td>1111</td>
</tr>
<tr>
<td>1</td>
<td>0010</td>
<td>1001</td>
</tr>
<tr>
<td>2</td>
<td>0100</td>
<td>0011</td>
</tr>
<tr>
<td>3</td>
<td>1000</td>
<td>0001</td>
</tr>
<tr>
<td>4</td>
<td>0011</td>
<td>0110</td>
</tr>
<tr>
<td>5</td>
<td>0110</td>
<td>1010</td>
</tr>
<tr>
<td>6</td>
<td>1100</td>
<td>0010</td>
</tr>
<tr>
<td>7</td>
<td>1011</td>
<td>0111</td>
</tr>
<tr>
<td>8</td>
<td>0101</td>
<td>1100</td>
</tr>
<tr>
<td>9</td>
<td>1010</td>
<td>1000</td>
</tr>
<tr>
<td>10</td>
<td>0111</td>
<td>0101</td>
</tr>
<tr>
<td>11</td>
<td>1110</td>
<td>1011</td>
</tr>
<tr>
<td>12</td>
<td>1111</td>
<td>0100</td>
</tr>
<tr>
<td>13</td>
<td>1101</td>
<td>1101</td>
</tr>
<tr>
<td>14</td>
<td>1001</td>
<td>1110</td>
</tr>
</tbody>
</table>
### B.3 $GF(2^5)$ using $p(x) = x^5 + x^2 + 1$ and $\alpha$ and $\beta \in GF(2^5)$

<table>
<thead>
<tr>
<th>Powers of $\beta$</th>
<th>Polynomial Basis $(\alpha^4 \alpha^3 \alpha^2 \alpha^1 \alpha^0)$</th>
<th>Normal Basis $(\alpha^{17} \alpha^{24} \alpha^{13} \alpha^{8} \alpha^{3})$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>000001</td>
<td>11111</td>
</tr>
<tr>
<td>1</td>
<td>00010</td>
<td>00011</td>
</tr>
<tr>
<td>2</td>
<td>00100</td>
<td>00110</td>
</tr>
<tr>
<td>3</td>
<td>01000</td>
<td>00001</td>
</tr>
<tr>
<td>4</td>
<td>10000</td>
<td>01100</td>
</tr>
<tr>
<td>5</td>
<td>00101</td>
<td>11001</td>
</tr>
<tr>
<td>6</td>
<td>01010</td>
<td>00010</td>
</tr>
<tr>
<td>7</td>
<td>10100</td>
<td>01010</td>
</tr>
<tr>
<td>8</td>
<td>01101</td>
<td>11000</td>
</tr>
<tr>
<td>9</td>
<td>11010</td>
<td>01110</td>
</tr>
<tr>
<td>10</td>
<td>10001</td>
<td>10011</td>
</tr>
<tr>
<td>11</td>
<td>00111</td>
<td>11010</td>
</tr>
<tr>
<td>12</td>
<td>01110</td>
<td>00100</td>
</tr>
<tr>
<td>13</td>
<td>11100</td>
<td>01011</td>
</tr>
<tr>
<td>14</td>
<td>11101</td>
<td>10100</td>
</tr>
<tr>
<td>15</td>
<td>11111</td>
<td>10111</td>
</tr>
<tr>
<td>16</td>
<td>11011</td>
<td>10001</td>
</tr>
<tr>
<td>17</td>
<td>10011</td>
<td>10000</td>
</tr>
<tr>
<td>18</td>
<td>00011</td>
<td>11100</td>
</tr>
<tr>
<td>19</td>
<td>00110</td>
<td>00101</td>
</tr>
<tr>
<td>20</td>
<td>01100</td>
<td>00111</td>
</tr>
<tr>
<td>21</td>
<td>11000</td>
<td>01101</td>
</tr>
<tr>
<td>22</td>
<td>10101</td>
<td>10101</td>
</tr>
<tr>
<td>23</td>
<td>01111</td>
<td>11101</td>
</tr>
<tr>
<td>24</td>
<td>11110</td>
<td>01000</td>
</tr>
<tr>
<td>25</td>
<td>11001</td>
<td>10010</td>
</tr>
<tr>
<td>26</td>
<td>10111</td>
<td>10110</td>
</tr>
<tr>
<td>27</td>
<td>01011</td>
<td>11101</td>
</tr>
<tr>
<td>28</td>
<td>10110</td>
<td>01001</td>
</tr>
<tr>
<td>29</td>
<td>01001</td>
<td>11110</td>
</tr>
<tr>
<td>30</td>
<td>10010</td>
<td>01111</td>
</tr>
</tbody>
</table>
APPENDIX C

WEIGHT DISTRIBUTION FOR RS CODES

C. 1 (7,3) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>1.470000e+02</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.470000e+02</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2.170000e+02</td>
<td></td>
</tr>
</tbody>
</table>

C. 2 (7,5) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2.450000e+02</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1.225000e+03</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.586000e+03</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>1.283800e+04</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.287300e+04</td>
<td></td>
</tr>
</tbody>
</table>

C. 3 (15,9) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>9.652500e+04</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>8.687250e+05</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>1.171170e+07</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>1.028828e+08</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>7.040534e+08</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>3.518547e+09</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1.218031e+10</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>2.610049e+10</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>2.610051e+10</td>
<td></td>
</tr>
</tbody>
</table>

C. 4 (15,11) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>4.504500e+04</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>8.258250e+05</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>1.689188e+07</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>2.514476e+08</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>2.936183e+09</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>2.642313e+10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>1.801594e+11</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>9.007962e+11</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>3.118141e+12</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>6.681731e+12</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>6.681731e+12</td>
<td></td>
</tr>
</tbody>
</table>

C. 5 (15,13) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>6.825000e+03</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2.661750e+05</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>8.918910e+06</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2.226724e+08</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>4.294880e+09</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>6.442262e+10</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>7.515977e+11</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>6.764379e+12</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4.612077e+13</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>2.306038e+14</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>7.982441e+14</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>1.710523e+15</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>1.710523e+15</td>
<td></td>
</tr>
</tbody>
</table>
## Appendix C. Weight Distribution for Reed-Solomon Codes

### C. 6 (31,15) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>8.220658e+09</td>
<td>24</td>
</tr>
<tr>
<td>18</td>
<td>9.590768e+10</td>
<td>25</td>
</tr>
<tr>
<td>19</td>
<td>2.629217e+12</td>
<td>26</td>
</tr>
<tr>
<td>20</td>
<td>4.676150e+13</td>
<td>27</td>
</tr>
<tr>
<td>21</td>
<td>7.646478e+14</td>
<td>28</td>
</tr>
<tr>
<td>22</td>
<td>1.076489e+16</td>
<td>29</td>
</tr>
<tr>
<td>23</td>
<td>1.305961e+17</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

### C. 7 (31,23) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>6.249623e+08</td>
<td>20</td>
</tr>
<tr>
<td>10</td>
<td>3.162309e+10</td>
<td>21</td>
</tr>
<tr>
<td>11</td>
<td>1.966006e+12</td>
<td>22</td>
</tr>
<tr>
<td>12</td>
<td>1.010520e+14</td>
<td>23</td>
</tr>
<tr>
<td>13</td>
<td>4.580544e+15</td>
<td>24</td>
</tr>
<tr>
<td>14</td>
<td>1.825609e+17</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>6.413989e+18</td>
<td>26</td>
</tr>
<tr>
<td>16</td>
<td>1.988336e+20</td>
<td>27</td>
</tr>
<tr>
<td>17</td>
<td>5.438685e+21</td>
<td>28</td>
</tr>
<tr>
<td>18</td>
<td>1.311327e+23</td>
<td>29</td>
</tr>
<tr>
<td>19</td>
<td>2.781394e+24</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

### C. 8 (31,25) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>8.151682e+07</td>
<td>19</td>
</tr>
<tr>
<td>8</td>
<td>6.113762e+09</td>
<td>20</td>
</tr>
<tr>
<td>9</td>
<td>4.974700e+11</td>
<td>21</td>
</tr>
<tr>
<td>10</td>
<td>3.385046e+13</td>
<td>22</td>
</tr>
<tr>
<td>11</td>
<td>2.003662e+15</td>
<td>23</td>
</tr>
<tr>
<td>12</td>
<td>1.035215e+17</td>
<td>24</td>
</tr>
<tr>
<td>13</td>
<td>4.690321e+18</td>
<td>25</td>
</tr>
<tr>
<td>14</td>
<td>1.869428e+20</td>
<td>26</td>
</tr>
<tr>
<td>15</td>
<td>6.567924e+21</td>
<td>27</td>
</tr>
<tr>
<td>16</td>
<td>2.036056e+23</td>
<td>28</td>
</tr>
<tr>
<td>17</td>
<td>5.569213e+24</td>
<td>29</td>
</tr>
<tr>
<td>18</td>
<td>1.342799e+26</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td></td>
<td>31</td>
</tr>
</tbody>
</table>

### C. 9 (31,27) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>5.267241e+06</td>
<td>10</td>
</tr>
<tr>
<td>6</td>
<td>6.162672e+08</td>
<td>11</td>
</tr>
<tr>
<td>7</td>
<td>6.904475e+10</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>6.416271e+12</td>
<td>13</td>
</tr>
<tr>
<td>9</td>
<td>5.083331e+14</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>15</td>
</tr>
</tbody>
</table>
### Appendix C. Weight Distribution for Reed-Solomon Codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>16</td>
<td>2.084922e+26</td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>5.702874e+27</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1.375026e+29</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2.916503e+30</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>5.424696e+31</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>8.808673e+32</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1.241222e+34</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1.505656e+35</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1.555845e+36</td>
<td></td>
</tr>
</tbody>
</table>

#### C. 10 (31,29) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>17</td>
<td>5.839743e+30</td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>1.40807e+32</td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>2.986499e+33</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>5.554889e+34</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>9.02008e+35</td>
<td></td>
</tr>
<tr>
<td>22</td>
<td>1.271011e+37</td>
<td></td>
</tr>
<tr>
<td>23</td>
<td>1.541792e+38</td>
<td></td>
</tr>
<tr>
<td>24</td>
<td>1.593185e+39</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>1.382885e+40</td>
<td></td>
</tr>
<tr>
<td>26</td>
<td>9.892945e+40</td>
<td></td>
</tr>
<tr>
<td>27</td>
<td>5.679283e+41</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2.515111e+42</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>8.065701e+42</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1.666912e+43</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>1.666912e+43</td>
<td></td>
</tr>
</tbody>
</table>

#### C. 11 (63,57) RS codes

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Weight</th>
<th>Number of Code words</th>
</tr>
</thead>
<tbody>
<tr>
<td>27</td>
<td>2.721661e+55</td>
<td></td>
</tr>
<tr>
<td>28</td>
<td>2.204545e+57</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1.676214e+59</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1.196817e+61</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>8.026396e+62</td>
<td></td>
</tr>
<tr>
<td>32</td>
<td>5.056630e+64</td>
<td></td>
</tr>
<tr>
<td>33</td>
<td>2.992605e+66</td>
<td></td>
</tr>
<tr>
<td>34</td>
<td>1.663537e+68</td>
<td></td>
</tr>
<tr>
<td>35</td>
<td>8.683661e+69</td>
<td></td>
</tr>
<tr>
<td>36</td>
<td>4.254994e+71</td>
<td></td>
</tr>
<tr>
<td>37</td>
<td>1.956147e+73</td>
<td></td>
</tr>
<tr>
<td>38</td>
<td>8.432024e+74</td>
<td></td>
</tr>
<tr>
<td>39</td>
<td>3.405240e+76</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>1.287181e+78</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>4.549085e+79</td>
<td></td>
</tr>
<tr>
<td>42</td>
<td>1.501198e+81</td>
<td></td>
</tr>
<tr>
<td>43</td>
<td>4.618803e+82</td>
<td></td>
</tr>
<tr>
<td>44</td>
<td>1.322657e+84</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>3.518268e+85</td>
<td></td>
</tr>
<tr>
<td>46</td>
<td>8.673296e+86</td>
<td></td>
</tr>
<tr>
<td>47</td>
<td>1.976404e+88</td>
<td></td>
</tr>
<tr>
<td>48</td>
<td>4.150449e+89</td>
<td></td>
</tr>
</tbody>
</table>
### Appendix C. Weight Distribution for Reed-Solomon Codes

<table>
<thead>
<tr>
<th>49</th>
<th>8.0044376e+90</th>
<th>57</th>
<th>3.609876e+99</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>1.4119836e+92</td>
<td>58</td>
<td>2.3526436e+100</td>
</tr>
<tr>
<td>51</td>
<td>2.2674786e+93</td>
<td>59</td>
<td>1.2560726e+101</td>
</tr>
<tr>
<td>52</td>
<td>3.2965646e+94</td>
<td>60</td>
<td>5.2755046e+101</td>
</tr>
<tr>
<td>53</td>
<td>4.3104136e+95</td>
<td>61</td>
<td>1.634541e+102</td>
</tr>
<tr>
<td>54</td>
<td>5.0288156e+96</td>
<td>62</td>
<td>3.3218106e+102</td>
</tr>
<tr>
<td>55</td>
<td>5.1842526e+97</td>
<td>63</td>
<td>3.3218106e+102</td>
</tr>
<tr>
<td>56</td>
<td>4.6658266e+98</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX D

ALGORITHM FOR SYSTEMATIC MATRIX

\[\text{Record the column of the first non-zero bit for each row} (P(i) \text{ for row } i)\]

\[i=0\]

\[\text{Yes}\]

\[(i,i)=1?\]

\[\text{No}\]

\[\text{Row Operation: } \text{row}(i) = \text{row}(0) + \text{row}(P(i))\]

\[i=i+1\]

\[i=k?\]

\[\text{Yes}\]

\[\text{Update the column of the first non-zero bit each row.} \]

\[i=0\]

\[\text{Delete non-zero bits within information block length } k\]

\[\text{for columns less than } column \ i \text{ by row operation.} \]

\[i=k?\]

\[\text{Yes}\]

\[i=0\]

\[\text{Delete non-zero bits within information block length } k\]

\[\text{for columns greater than } column \ i \text{ by row operation.} \]

\[i=k?\]

\[\text{Yes}\]

\[\text{Stop}\]

\[b(i,i): \text{a bit of row } i \text{ and column } i,\]

\[P(i): \text{column of the first non-zero bit for each row}\]
References


References


