

**A STUDY OF ERASURE CORRECTING CODES**

by

**Jing Cai**

A thesis submitted to the University of Plymouth  
in partial fulfilment for the degree of

**DOCTOR OF PHILOSOPHY**

School of Computing and Mathematics  
Faculty of Technology



**July 2010**

!!

---

9008778607  
THESIS 005.1 CAL

# A Study of Erasure Correcting Codes

by

Jing Cai

## Abstract

This work focus on erasure codes, particularly those that of high performance, and the related decoding algorithms, especially with low computational complexity. The work is composed of different pieces, but the main components are developed within the following two main themes.

Ideas of message passing are applied to solve the erasures after the transmission. Efficient matrix-representation of the belief propagation (BP) decoding algorithm on the BEC is introduced as the recovery algorithm. Gallager's bit-flipping algorithm are further developed into the guess and multi-guess algorithms especially for the application to recover the unsolved erasures after the recovery algorithm. A novel maximum-likelihood decoding algorithm, the In-place algorithm, is proposed with a reduced computational complexity. A further study on the marginal number of correctable erasures by the In-place algorithm determines a lower bound of the average number of correctable erasures. Following the spirit in search of the most likable codeword based on the received vector, we propose a new branch-evaluation-search-on-the-code-tree (BESCT) algorithm, which is powerful enough to approach the ML performance for all linear block codes.

To maximise the recovery capability of the In-place algorithm in network transmissions, we propose the product packetisation structure to reconcile the computational complexity of the In-place algorithm. Combined with the proposed product packetisation structure, the computational complexity is less than the quadratic complexity bound. We then extend this to application of the Rayleigh fading channel to solve the errors and erasures. By concatenating an outer code, such as BCH codes, the product-packetised RS codes have the performance of the hard-decision In-place algorithm significantly better than that of the soft-decision iterative algorithms on optimally designed LDPC codes.

---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	An Overview of Coding Theory in Recent Years . . . . .	1
1.2	Basics of a Communication System . . . . .	4
1.2.1	Source Encoder and Encryption Encoder . . . . .	4
1.2.2	Channel Encoder . . . . .	5
1.2.3	Modulation . . . . .	6
1.2.4	Channel . . . . .	6
1.2.5	Demodulation . . . . .	7
1.2.6	Channel Decoder . . . . .	7
1.3	A Development of Erasure Correcting Codes . . . . .	7
1.3.1	Reed-Solomon Codes: Optimal Erasure Codes . . . . .	9
1.3.2	Fountain Codes: Sub-Optimal Erasure Codes . . . . .	11
1.4	Basic Definitions . . . . .	12
1.5	Thesis Contributions and Outline . . . . .	14
1.5.1	Contributions . . . . .	14
1.5.2	Thesis Outline . . . . .	16
<b>I</b>	<b>Erasure Decoding Algorithms</b>	<b>21</b>
<b>2</b>	<b>Matrix-based Erasure Decoding Algorithms</b>	<b>23</b>
2.1	Background . . . . .	23
2.2	A Review of the Recovery algorithm . . . . .	24

## CONTENTS

---

2.3	Matrix-based Iterative Decoding Algorithm and its Variances via the BEC . . . . .	28
2.3.1	Matrix Representation of the Erased Bits . . . . .	28
2.3.2	Matrix-based Iterative Recovery Algorithm . . . . .	30
2.3.3	Guess/Multi-Guess Algorithm . . . . .	35
2.4	Matrix-based In-place Decoder . . . . .	41
2.5	Numerical Results and Discussion . . . . .	48
2.5.1	Performance of the Recovery Algorithm . . . . .	48
2.5.2	Performance of the Guess/Multi-Guess Algorithms . . . . .	49
2.5.3	Performance of the In-place Algorithm . . . . .	55
2.6	Summary . . . . .	59
<b>3</b>	<b>Branch-Evaluation Search on the Code-Tree Algorithm</b>	<b>61</b>
3.1	Background . . . . .	61
3.2	A Review of the Ordered Statistical Decoding Algorithm . . . . .	63
3.3	Code Tree Representation of Linear Block Codes . . . . .	65
3.4	BESCT Algorithm . . . . .	68
3.4.1	The Concept of Ensemble Branch . . . . .	71
3.4.2	Cost and Dynamic Threshold . . . . .	72
3.4.3	The BESCT Algorithm . . . . .	74
3.5	Numerical Results and Discussion . . . . .	76
3.5.1	Performance of the BESCT algorithm . . . . .	76
3.5.2	Discussion on the Computational Complexity of the BESCT Algorithm . . . . .	83
3.6	Summary . . . . .	86
 <b>II Packet Data Transmission and Coding Arrangement</b>		
<b>87</b>		
<b>4</b>	<b>The Packet Data Transmission System of Fountain Codes</b>	<b>89</b>
4.1	Background . . . . .	89
4.2	LT Codes . . . . .	90
4.2.1	LT Encoding Process . . . . .	91

4.2.2	LT Decoding Process . . . . .	93
4.2.3	Degree Distribution $\rho(d)$ . . . . .	94
4.3	Raptor Codes . . . . .	98
4.3.1	Non-Systematic Raptor Codes . . . . .	99
4.3.2	Systematic Raptor Codes . . . . .	100
4.4	Numerical Results and Discussion . . . . .	103
4.5	Summary . . . . .	111
<b>5</b>	<b>Capacity Approaching Codes for the Binary Erasure Channel Using a Product Packetisation Method</b>	<b>113</b>
5.1	Background . . . . .	113
5.2	Product-packetisation Transmission Protocol . . . . .	115
5.2.1	Product Packetisation and De-packetisation . . . . .	115
5.2.2	Protocol Description . . . . .	117
5.3	Rateless Reed Solomon Codes . . . . .	118
5.4	ML Decoding Algorithm on Rateless RS Codes . . . . .	120
5.5	Numerical Results and Discussion . . . . .	122
5.6	Summary . . . . .	128
<b>6</b>	<b>Concatenated Reed-Solomon Coding with Hard-decision for the Rayleigh Fading Channel</b>	<b>129</b>
6.1	Background . . . . .	129
6.2	System and Channel Model . . . . .	130
6.2.1	Rayleigh Distribution . . . . .	130
6.2.2	Rayleigh Fading Channel . . . . .	133
6.3	the System Arrangement . . . . .	133
6.4	Analysis of Concatenated RS Codes . . . . .	135
6.5	Numerical Results and Discussion . . . . .	136
6.6	Summary . . . . .	140
<b>7</b>	<b>Conclusions and Future Work</b>	<b>141</b>
7.1	Conclusions . . . . .	141
7.2	Future Work . . . . .	145

## CONTENTS

---

References	154
A Publications	155



## List of Figures

1.1	A General Construction of a Digital Communication System . . .	5
2.1	the Recovery Algorithm over the Bipartite Graph: square spots representing the check vertex and circle spots denoting the variable vertex . . . . .	25
2.2	an Example of a Bipartite Expression of a Code . . . . .	26
2.3	Matrix-based Iterative Recovery Decoder . . . . .	33
2.4	The BER/FER performance of the PEG LDPC code (256,128) with the Matrix-based Recovery Algorithm . . . . .	34
2.5	Correction of Erased Bits Using Guess Algorithm . . . . .	36
2.6	Performance of the PEG LDPC (256,128) with the Recovery Algorithm and C-Guess Algorithm . . . . .	37
2.7	Performance Comparison between the Crucial Guess algorithm (C-Guess) and the Sequential Guess algorithm (S-Guess) of the PEG LDPC (256,128) . . . . .	38
2.8	Matrix-based In-place Algorithm . . . . .	43
2.9	A performance comparison on the results of the Recovery decoder, Guess decoder (including C-Guess and S-Guess) and the In-place decoder . . . . .	47
2.10	Performance of the LT codes and irregular LDPC codes with the erasure probability $p = 0.2$ . . . . .	48
2.11	Performance of the Cyclic LDPC (255,175) with the Guess and In-place algorithm . . . . .	50
2.12	Different guess algorithms for the PEG LDPC (256,128) . . . . .	52

## LIST OF FIGURES

---

2.13	Different guess algorithms for the LDPC (1024,512,24) . . . . .	53
2.14	Different guess algorithms for the OSML(255,175) . . . . .	54
2.15	Performance of the Cyclic LDPC (341,205) with the Recovery, the Guess, the Multi-Guess and the In-place Algorithms . . . . .	55
2.16	Comparison of Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $n - k$ ) for (255,175) code and BCH (255,178) code . . . . .	56
2.17	Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $n - k$ ) for (103, 52) code quadratic residue code . . . . .	57
2.18	Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $n - k$ ) for (128,64) extended BCH code	58
3.1	An example of Code Tree I (One-directional Code Tree) . . . . .	67
3.2	An Example of Code Tree II (Bi-directional Code Tree) . . . . .	68
3.3	$\mathcal{T}_{BESCT}$ of the Hamming code(7,4,3). A coding rule has to be applied to validate a codeword. . . . .	71
3.4	Performance of the PEG-LDPC code (256, 128, 17) . . . . .	78
3.5	Performance of the EG-LDPC code (63, 37, 9) . . . . .	79
3.6	Performance of the OSML code (255, 175) . . . . .	80
3.7	the Frame-error-rate performance of the LQR (104, 52, 20) . . . . .	82
3.8	The Complexity Comparison between the OSD algorithm and the BESCT algorithm . . . . .	85
4.1	LT Encoding Processing . . . . .	92
4.2	the Degree Distribution Generated from the Ideal Soliton Distri- bution for a LT code with information length of 15 . . . . .	96
4.3	Degree Distributions based on Different Methods, $Pkt_{no}(info) =$ 1024 . . . . .	98
4.4	the Graphical Expression of a non-systematic Raptor Encoding Process . . . . .	99
4.5	Systematic Raptor encoder . . . . .	102
4.6	Histogram of the actual number of required transmission packets when the number of information packets is 10,000 and $p = 0.5$ . . .	104

## LIST OF FIGURES

---

4.7	Histogram of the actual number of required transmission packets when the number of information packets is 5000 and $p = 0.5$ . . .	105
4.8	Histogram of the actual number of required transmission packets when the number of information packets is 1024 and $p = 0.5$ . . .	106
4.9	Histogram of the actual number of required transmission packets when the number of information packets is 512 and $p = 0.5$ . . . .	107
4.10	Histogram of the actual number of required transmission packets when the number of information packets is 250 and $p = 0.5$ . . . .	108
4.11	Histogram of the actual number of required transmission packets when the number of information packets is 125 and $p = 0.5$ . . . .	109
4.12	A Comparison on the Raptor Code Performances with Different Pre-encoding Scheme: Binary Reed-Solomon Codes vs Regular LDPC Codes . . . . .	110
5.1	Packet construction for the $i$ -th packet . . . . .	116
5.2	The Protocol Structure . . . . .	117
5.3	FER vs the erasure probability of erased packets for the cyclic code (225, 105) and the LT (225, 105) code over the Erasure Channel, both using ML decoding . . . . .	123
5.4	FER vs the erasure probability of erased packets for the cyclic code (3969,2016) based on the BCH(63,32,12) and the Reed Solomon(63,32,32), the LT(3969,2016) code and theoretical result over the Erasure Channel . . . . .	125
5.5	Histogram of the cyclic code (3969,2016) based on the BCH(63,32,12) when $p = 0.25$ , mean = 44.7 . . . . .	125
5.6	Histogram of the cyclic code(3969, 2016) based on the RS(63, 32, 32) when $p = 0.25$ , mean = 42.7 . . . . .	126
5.7	FER vs the erasure probability of erased packets for the cyclic code (76500,39300) based on the BCH (255, 131, 37) and the RS (255, 131, 125) over the Erasure Channel . . . . .	126
5.8	Histogram of the cyclic code(76500,39300) based on the BCH(255, 131, 37) when $p = 0.42$ , mean = 227.95 . . . . .	127

## LIST OF FIGURES

---

5.9	Histogram of the cyclic code(76500,39300) based on the Reed-Solomon(255,131,125) when $p = 0.42$ , mean = 225.86 . . . . .	127
6.1	The PDF of the Rayleigh random variables for different values of $\sigma$	132
6.2	The CDF of the Rayleigh random variables for different values of $\sigma$	132
6.3	an Overview of the Proposed Transmission System . . . . .	134
6.4	RS codes with variable code rates in the Rayleigh fading channel .	137
6.5	RS (63, 59, 5) concatenated by different Hamming or BCH codes in the Rayleigh fading channel . . . . .	138
6.6	RS with BCH hard-decision codes vs. LDPC soft-decision codes in the Rayleigh fading channel . . . . .	139

## List of Tables

3.1	Partial Weight Enumerators Estimated by the TCSE Algorithm [2]	
	(A) . . . . .	77
3.2	Partial Weight Enumerators Estimated by the TCSE Algorithm [2]	
	(B) . . . . .	78
4.1	Shokrollahi Degree Distribution [69] . . . . .	97
4.2	The Overhead Percentages on the different sizes of the information files, when $p = 0.5$ . . . . .	107

## LIST OF TABLES

---

## List of Algorithms

2.1	RecoveryDec( $n, \mathbf{H}, \mathbf{y}$ ) . . . . .	31
2.2	Polynomial-update Algorithm . . . . .	42
4.1	LTEnc( $\rho(\cdot), \mathbf{x}, \mathbf{u}$ ) . . . . .	92
4.2	LTDec-XOR( $\mathbf{y}, \rho(d), \mathbf{s}$ ) . . . . .	94

To My Beloved Family:  
My Mother, Yifen Wei, My Father, Mingsong Cai

and

To the Wonderful World.

*Look deep, deep into nature, and then you will understand everything better.*

-- Albert Einstein



## Acknowledgements

It is a great pleasure to take this opportunity to thank all the people who have helped my biggest dream of this thesis come true. I am very fortunate to have Prof. Martin Tomlinson as the supervisor of my PhD study. It was his wonderful lectures on digital communications and coding theory that first attracted me into this field. During this long-time study, I have encountered several ups and downs where I struggled to grow personally and professionally. Had it not been for his brilliant insights, consistent guidance, constant encouragement, friendly advice and enormous support, this work could not have been completed. For all the things I have learnt from him, I will always be indebted to him.

My heartfelt thanks to my other supervisors, Dr. Marcel Ambroze and Dr. Mohammed Zaki Ahmed. Discussions with them are always informative and fruitful. There was never a question they could not answer, never a problem that they could not solve. They are definitely some of my favourite people in Plymouth.

I am incredibly grateful to Dr. Cen Tjhai for numerous helpful and fun discussions. A hearty thank you also goes to all my current and former group members, Cen Tjhai, Evangelos Papagiannis, Purav Shah, Xin Xu, Andrew Rogers, Ismail Isnin, Keiko Takeuchi, Marco Domes, Mubarak Jibril and Li Yang. Sharing with them the joy and frustration has made the life of the PhD beautiful.

I would like to thank the design team, John Parker-Smith, Kevin

Steward and Matthew Radmore, at Bombardier Transportation UK Ltd, for proof-reading parts of this thesis. My thanks also go to Dr. Mohammed Zaki Ahmed for the corrections on the thesis.

My thanks also go to my managers, Mike Mackie and Andy Millar at Bombardier Transportation UK Ltd. Thanks for their understanding and flexible time arrangements for my study.

Li Yang, thank you for keeping me sane in last two years. Words cannot express how grateful I am to you for your unending confidence in me.

Lastly but not leastly, I would like to thank my family. My parents, Mingsong Cai and Yifen Wei, taught me those most valuable "theorems of life" which I could not learn from any text book or paper. Thanks to my grandma, who firstly taught me the math when I was three.

## AUTHOR'S DECLARATION

At no time during the registration for the degree of Doctor of Philosophy has the author been registered for any other University award without prior agreement of the Graduate Committee.

This study was financed by the Dean of Technology Award.

A programme of advanced study was undertaken, which included the extensive reading of literature relevant to the research project and attendance of international conferences on coding and communications.

Relevant scientific seminars and conferences were regularly attended at which work was often presented; external institutions were visited for consultation purposes and several papers prepared for publication.

Publications (or presentation of other forms of creative and performing work):

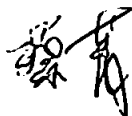
- Tomlinson, M., Tjhai, C., Cai, J. and Ambroze, M., "Analysis of the Distribution of the Number of Erasures Correctable by a Binary Linear Code and the Link to Low Weight Codewords", *IET Proceedings Communications* 1(3):539-548, June 2007.
- Cai, J., Tomlinson, M., Tjhai, C., Ambroze, M., Ahmed, M., "Comparison of Concatenated Reed-Solomon Coding with Hard-decision to Soft-decision LDPC Coding for the Rayleigh Fading Channel", in Proc. IEEE Information Theory Workshop, Chengdu, China, 22-26 Oct., 2006.
- Cai, J., Tomlinson, M., Tjhai, C., Ambroze, M. and Ahmed, M., "Advances in Iterative Decoding and Maximum Likelihood Decoding for the Packet Network with Comparisons to Fountain

Codes over the Erasure Channel”, *in* Proc. 4th International Symposium on Turbo Codes in connection with 6th International ITG-Conference on Source and Channel Coding, Munich, Germany, 3-7 April, 2006.

- *Cai, J.*, Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M., “An Efficient Solution to Packet Loss: Erasure Correcting Codes”, *in* Proc. 4th IASTED International Conference Communication Systems and Networks, Benidorm, Spain, 12-14 Sep., 2005.
- *Cai, J.*, Ambroze, M., “Analysis on Decoding Algorithms Based on Sectionalised Trellises of Block Codes and Their Dual”, *in* Proc. PREP, Lancaster, 30 Mar-1 April, 2005.
- Tomlinson, M., *Cai, J.*, Tjhai, C., Ambroze, M. and Ahmed, M., “System for Correcting Deleted or Missing Symbols”, UK Patent Application, GB 2421599, 28 June, 2006.

Word count of main body of thesis: 56901

Signed



Date

27/10/2010

## 1.1 An Overview of Coding Theory in Recent Years

---

In the first few decades, algebraic codes dominated error-controlling technology, which aimed to approach the optimum detecting/decoding performance by maximising the minimum Hamming distance (*Definition 1.6*) within a reasonable scope restricted by the code's information length. In 1954, Muller [46] and Reed [58] both presented the *Reed-Muller* (RM) code and its efficient decoding algorithm. By that time, it was realised that with the hard-decision decoding algorithm (HDD), it is hard to reach the Shannon limit and therefore a so-called soft-decision decoding algorithm (SDD) based on the received floating-point data has been proposed and developed for the modern coding scheme. Also in 1954, Silverman & Balser [70] first described an SDD algorithm – the *Wagner decoding algorithm*. Since then, the reliability of received channel output has been gradually taken into account to consummate the idea of a soft-decision decoding algorithm. The sixties was the dominative period of cyclic codes [53], which were inspired by the RM code. The major property of cyclic codes is the invariance of arbitrary cyclic shifts of their codewords. With this particularly cyclic property, Hocquenghem [30] and Bose & Ray-Chaudhuri [8] invented the well-known *Bose-Chaudhuri-Hocquenghem* code (BCH) over  $GF(2)$  (*Definition 1.9*) with a specified design Hamming distance. Also in 1960, Reed and Solomon [59] produced the eponymous *Reed Solomon* codes (RS), which is a class of non-binary BCH codes. Decoding algorithms using finite-field arithmetic were then developed by Peterson [52]. Berlekamp [4] introduced a fast algorithm for BCH/RS decoding and later the following year, Massey [45] further reduced the decoding complexity of the algorithm. The combined *Berlekamp-Massey* algorithm became the standard for the following decade. Another approach to decode block codes by adapting the channel measurement information was introduced by Chase [13].

Another branch of code development was initialised by Shannon's probabilistic approach to coding, which were called *convolutional codes* invented by Elias [19]. Using the designed structure of convolutional codes, a sequential search decoding algorithm was proposed by Wozencraft & Reiffen [79]. A breakthrough invention to decode convolutional codes "asymptotically optimal" was made by Viterbi [77]. The *Viterbi algorithm* employs the received soft decision data which shows a new approach to compute the *a posteriori probability* (APP) based on the reliability information. Before the invention of the Viterbi algorithm, Gallager [24] had

## 1. INTRODUCTION

---

used the APP methodology with an iterative message-passing structure for the decoding of his proposed low-density parity-check codes (*LDPC*). Also, a similar idea of the APP decoding algorithm had also been utilised to unify the study of majority logic decoding, known as the *threshold decoding algorithm* [44].

During the last 20 years, the probabilistic coding theory has stepped up to a new stage. Turbo codes, invented by Berrou *et al.* [6] in 1993, and LDPC codes, resuscitated by MacKay & Neal [42] in 1996, are codes defined on graphs with iterative probabilistic decoding algorithms. Both of them are random enough to come very close to capacity of the channel, but constructive enough to be iteratively decodable in polynomial time.

With Shannon's promise, for every channel, there are error-correcting codes of rate up to the channel capacity that achieve probability of error as small as the users want. In sixty years, Shannon's idea "A Mathematical Theory of Communications" has successfully been developed into practice, and Shannon's puzzle on such codes and their decoding algorithms has motivated enormous amount of coding research.

### 1.2 Basics of a Communication System

To appreciate the contributions of error-correcting coding technology and understand the coding limitations, a knowledge of digital communication systems is required. A straightforward description of a communication system is passing information from a source (transmitter) to a sink (receiver) via a transmission medium (channel).

Figure 1.1 illustrates a general construction of a single digital communication system. In this system, a digital signal from a source is encoded, modulated, demodulated and then decoded to a sink.

#### 1.2.1 Source Encoder and Encryption Encoder

##### ✓ *Source Encoder*

Unless the source signal is already in digital form, any kind of source to be transmitted has to be converted by an analog-to-digital process. The source encoder is

*Information is the resolution of uncertainty.*

Claude Shannon (1916-2001)

# 1

## Introduction

### 1.1 An Overview of Coding Theory in Recent Years

In the last 200 years, people's lives have been digitised since the first telephone bell rang in 1844. The remarkable work done by Nyquist [47] has been considered as the stem of modern digital communications, which was the first time to formulate the maximum theoretical baud rate for a band-limited channel. The famous *Nyquist Rate* declares the ISI (*inter-symbol interference*) free baud rate (also called as *symbol rate*) as  $2W$  when the pulse is shaped perfectly by the sinc function \*, where  $W$  is the bandwidth of the transmission channel. Coupling with Nyquist's work, Hartley [29] investigated the relationship between the maximum signal amplitude  $A_{\max}$  and the amplitude resolution  $A_\delta$ . If considering that the amplitude resolution  $A_\delta$  partitions the signal as a binary sequence composed by bits, in order to maintain a reliable distinguishing among the partitions, the

---

\* $\text{sinc}(x) = \frac{\sin(x)}{x}$

## 1. INTRODUCTION

---

maximum transmission rate  $\mathcal{R}$  is

$$\mathcal{R} = 2W \log_2 \left( 1 + \frac{\mathcal{A}_{\max}}{\mathcal{A}_\delta} \right) \text{bits/second} \quad (1.1)$$

Enlightened by Nyquist's and Hartley's work on the maximum transmission rate, Claude Shannon's remarkable work "*A Mathematical Theory of Communication*" [66] developed the twin disciplines of *information theory* and *coding theory*. Generally speaking, information theory attempts to study achievable bounds for communication and is largely probabilistic and analytic in nature, and coding theory works to realise the promise of models which are constructed through mainly algebraic means. In this pioneering work, Shannon pointed out that the channel noise need not cause any degradation in the transmission reliability which can be realised by employing error-correcting codes. Based upon his statistical analysis on information sources and communication channels, Shannon provided a novel parameter, the *channel capacity*  $\mathbf{C}$  which is associated with the effect of a transmitter power constraint, a bandwidth constraint, and additive noise. Under the assumption of an additive white Gaussian noise (AWGN) disturbance, the band-limited transmission channel has its channel capacity  $\mathbf{C}$  as

$$\mathbf{C} = W \log_2 \left( \frac{P}{WN_0} \right) \text{bits/second} \quad (1.2)$$

where  $P$  is the average transmitted power and  $N_0$  is the power spectral density of the additive noise. The channel capacity formula basically gives the constraint on the information rate  $\mathcal{R}$ , that is

- ( $\mathcal{R} \leq \mathbf{C}$ ) An error-free transmission is theoretically established by appropriate coding.
- ( $\mathcal{R} > \mathbf{C}$ ) It is impossible to realise an accurate transmission.

In 1950, in order to solve a small number of errors on magnetic storage media, Hamming introduced *error-correcting codes* and in his milestone paper [28] gave the description of *Hamming Codes*. Combining Hamming's work with Shannon's information theory, the era of error-correcting (controlling) coding theory was started.



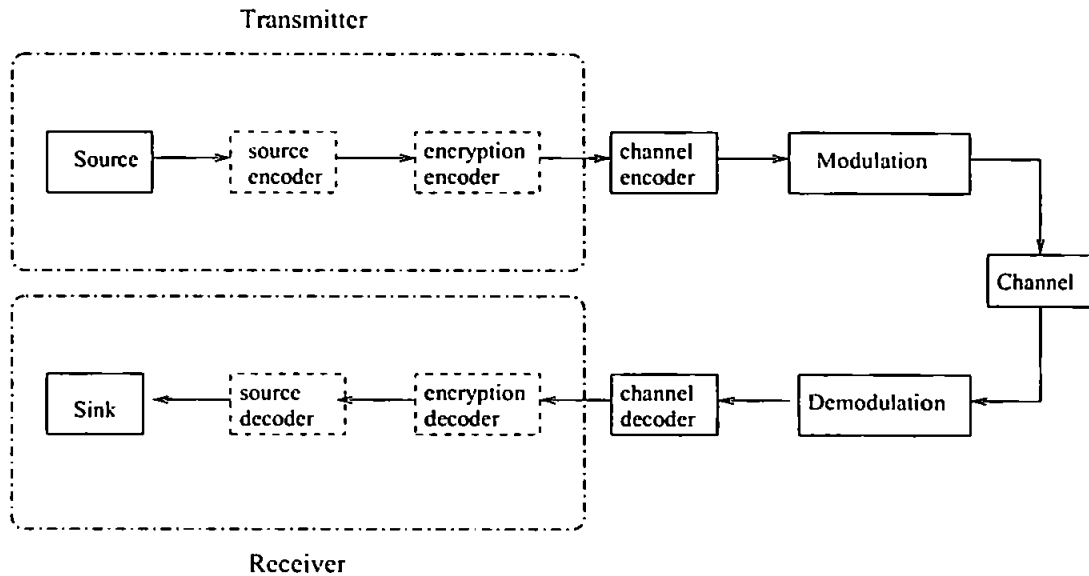


Figure 1.1: A General Construction of a Digital Communication System

used to compress the digitised source signal in case the source exceeds the number of bits of actual information content. The coding technique applied to the source coding is so-called “data compression codes”, such as Huffman code, Lempel-Ziv code, which are not in the scope of this thesis.

### ✓ Encryption Encoder

After the source encoding, the encrypter protects the information content by transforming or scrambling information into a data sequence which is unreadable to anyone except those possessing special knowledge, usually referred to as a key. Conceptually, the coding technique for the encryption is different to that for the error controlling coding. Also, it is not a necessary part of a communication system which falls beyond the scope of this thesis.

## 1.2.2 Channel Encoder

The channel encoder, as the first step in the error controlling process, introduces the *parity-check* (also called the *redundancy*) padding into the informa-

## 1. INTRODUCTION

---

tion stream. Following the definitions in many publications [43][37], the channel encoder generates an  $n$ -length data sequence dependent on every  $k$ -length information sequence, where  $n > k$ . The rate of the channel encoder is called the *code-rate*, denoted as  $R$ .

$$R = k/n \quad (1.3)$$

The information sequence is referred to as *information symbols* or *information bits* if the data is in binary form. Similarly for the parity-check sequence generated by the encoder, it is referred to as *parity-check symbols* or *parity-check bits*.

### 1.2.3 Modulation

Before sending the encoded data sequence into the channel, the data sequence needs converting into a suitable signal and this process is called *modulation*. The process of modulation is to map a stream of bits, containing 0s and 1s, into waveform. The  $z$ -Phase-Shift Keying ( $z$ PSK) modulation has been widely used, in which when  $z = 2$  or  $4$ , it is known as *Binary Phase Shift Keying (BPSK)* or *Quadrature Phase Shift Keying (QPSK)* modulation. In this thesis, we assume all the transmitted data sequences are after the BPSK modulation process.

### 1.2.4 Channel

The channel is the medium where the information symbols are conveyed. Except for conventional channels of two or more geographical points of communication such as telephone lines, Internet cables, radio channels, optical lines and so on, the channels of two or more different time points of communications also employ the channel coding, such as hard drives, CD-ROMs, DVDs and so on. It should be mentioned that the channel is *discrete*, and henceforth only finite alphabets are considered in this thesis. The conventional channel models include the *binary symmetric channel (BSC)* and the *additive white Gaussian noise channel (AWGN)*. However, the deficiency of the conventional channel models is that in reality, the transmission conditions and environmental noise are also changeable.

### 1.3 A Development of Erasure Correcting Codes

---

Therefore, conventional analysis offers a strong temptation to model the interference as some additional additive level of AWGN noise, which seldom yields meaningful results for real applications. Many real communication channels contain both AWGN noise and non-AWGN interference. If the AWGN noise is relatively small compared with the interference, and then we neglect the white Gaussian noise and only consider the interference which is detectable. By blanking all the detected interference, the channel model is called *the binary erasure channel* (BEC). If the AWGN noise causes transmission errors and also the fading on the signal power leads detectable empty slots, the channel model is called a *fading channel*.

In this thesis, we focus on the application of erasure codes for the BEC, the AWGN channel and the Rayleigh fading channel.

#### 1.2.5 Demodulation

The received signal from the channel is converted into a sequence of bits/symbols. Normally, a demodulator include the functions of filtering, demodulation, local time/frequency synchronisation and matched filtering. In this thesis, we suppose all the received signal being synchronised and demodulated before the decoding process.

#### 1.2.6 Channel Decoder

The decoder at receiver is to exploit the parity checks produced by the channel encoder to correct any errors or recover any erasures that may have been introduced.

### 1.3 A Development of Erasure Correcting Codes

Nowadays, Internet communications have become one of the most popular communications. People can send emails, images, sounds and videos to anyone anywhere in world through the Internet. Especially, the launch of YouTube, a video sharing website, has unintentionally affected people's daily life globally. The Internet has gradually taken place of the conventional entertainments, business and

## 1. INTRODUCTION

---

communications. Henceforth, to ensure a network communication reliable, highly efficient and low-cost has become of a great importance.

The channel model for network transmission is the *binary erasure channel* (BEC), introduced by Elias [20] in 1956. The transmitted data are catalogued as the non-erasure and the erasure only.

*Erasure codes* is a special case of error-correcting codes, which is known as a kind of forward error correction codes especially for the erasure channel. Define the code  $\mathcal{C}$  to be an  $(n, k)$ , where  $n$  is the code length and  $k$  is the information length.

On network transmissions, data is transmitted in the form of packets. Each packet includes a header section and a data section. The header contents are different according to different network protocols. Generally, it gives a description of the information (source) and the destination of the packet. The data section contains the information with its typical length from 30 bits to 1000 bits for most multicasting and broadcasting systems.

The erasure correcting performance of codes and associated decoding algorithms has received renewed interest in the study of coding over packet networks as a means of providing efficient computer communication protocols [61].

By employing the code  $\mathcal{C}(n, k)$ ,  $k$  information packets are encoded into  $n$  packets, which are called as the *transmission packets*. The erasure code  $\mathcal{C}$  is capable of recovering the information file of the size of  $k$  packets from a subset of  $n$  transmission packets. As defined for a linear block code, the fraction of  $k/n$  is called the code rate. In the transmission of the erasure channel, there is another fraction  $k'/k$  where  $k'$  is the actual number of the transmission packets required to recover  $k$  information packet, which is very important to identify the *efficiency* of the applied erasure coding scheme.

Erasure codes are catalogued into two types:

- *Optimal erasure codes* have the property to recover  $k$  information packets after receiving  $k$  distinct encoded packets.
- *Sub-optimal erasure codes* are capable of recovering  $k$  information packets when  $k(1 + \epsilon)$  encoded packets have been received, where  $\epsilon$  is a positive number related to the channel condition ( $0 \leq \epsilon \leq 1$ ).

### 1.3.1 Reed-Solomon Codes: Optimal Erasure Codes

*Reed-Solomon* (RS) codes were developed in 1960 by Reed & Solomon [57], and the breakthrough work was presented as a seminal article as entitled as “*Polynomial Codes over Certain Finite Fields*”. Later in 1969, an efficient decoding algorithm, the Berlekamp-Massey decoding algorithm, was invented by Berlekamp [4], Massey [44]. In 1977, RS codes were notably implemented in the Voyager program in the form of concatenated codes. The first commercial application in mass-produced consumer products appeared in 1982 with the compact disc, where two interleaved RS codes are used. Nowadays, RS codes, as one of the most powerful classic codes, have been widely implemented in digital storage devices and digital communication standards, such as in CDs, DVDs, Blu-ray Discs, in data transmission technologies such as DSL and WiMAX, in broadcast systems such as DVB and ATSC, and in computer applications such as RAID 6 systems.

RS codes belong to non-binary BCH code family and therefore they have the *cyclic* property as BCH codes. A RS  $\mathcal{C}_{RS}(n, k, d)$  over the Galois Field  $GF(2^m)$  has the parameters as in (1.4)

$$d = n - k + 1, \text{ where } k < n < 2^m + 2 \quad (1.4)$$

Normally, a RS code has its code length of  $n = 2^m - 1$ . A *doubly-extended RS* code can be constructed with its code length of  $n = 2^m$  or  $n = 2^m + 1$ . There also exist  $(2^m + 2, 3, 2^m)$  and  $(2^m + 2, 2^m - 1, 4)$  *triply-extended RS* codes. [43]

As an erasure code, it can recover up to  $d - 1$  known erased symbols. Also, for a noisy channel, it can detect and correct combinations of errors and erasures. RS codes are non-binary codes and each element of a RS codeword is called as a *symbol*. Therefore, a symbol-information sequence is viewed as a set of coefficients of a polynomial  $v(x)$  over a finite field. Because the RS code are a special case of a cyclic non-binary BCH code, the encoding process is equivalent to a derivation process from the coefficients of the information polynomial  $v(x)$  with a cyclic generator or parity-check polynomial.

To decode an RS code in noisy channel, there are mainly three types of decoding approaches.

## 1. INTRODUCTION

---

- *Algebraic Decoding* (also known as *Bounded Distance Decoding*): The algebraic decoders are capable of decoding up to  $t_0 = \frac{n-k}{2}$  error symbols, which generally follow the steps:

1. Computation of the syndrome.
2. Determination of an *error locator polynomial*.<sup>†</sup>

There are several different methods to identify the locator polynomial, including the well-known Berlekamp-Massey (BM) algorithm [4, 45], the Euclidean algorithm and so on.

3. Identifying the roots of the error locator polynomial.

Normally, it is done by applying the Chien search [14] which is an exhaustive search over all the elements in the field.

4. Determination of error values. Typically, it is accomplished by deploying Forney's algorithm [22].

- *Maximum Likelihood (ML) Decoding*: The ML decoders choose the codeword which is closest to the given received vector. However, the ML decoding is computationally difficult in general in the AWGN channel. For the application in the BEC, the ML decoding is equivalent to solving the linear equation between the parity-check matrix and the received vector. Our proposed algorithm, the In-place algorithm, is a complexity-reduced ML decoding algorithm for the BEC.

- *List Decoding*: The list decoders is to find *all* potential codewords which are within a given distance of the received vector.

The idea of list decoding was initialised by Elias [21]. The recent Guruswami-Sudan (GS) algorithm [26] provides lists of all codewords within a given distance of the received vector and is able to correct up to  $t_{GS} = \lfloor n - \sqrt{nk} - 1 \rfloor$  errors.

In the recent several years, soft-decision decoding algorithms for RS codes have been the topic of significant research interest. Koetter and Vardy [34] extended the GS algorithm to an algebraic soft-decision decoding algorithm, called

---

<sup>†</sup>The error locator polynomial indicates the error locations with its roots.

as the KV algorithm, which significantly outperforms the list decoding algorithm. However, as a development of the GS list algorithm, the KV algorithm suffers an immensely high computational complexity. Therefore, multiple runs of errors-and-erasures or error-only decoding with some low complexity algorithm, e.g. the BM algorithm, has been re-studied in [3, 36, 80]. The successive decoding algorithm with multiple thresholds is capable of performing as the KV algorithm with almost the same average complexity as a conventional hard-decision decoder.

#### 1.3.2 Fountain Codes: Sub-Optimal Erasure Codes

Fountain Codes, known as a class of sub-optimal erasure codes, have been defined by their property of *rateless* generation. The term *rateless* literally claims the fact that these codes do not exhibit a fixed code rate. For  $k$  information packets, potentially limitless transmission packets can be encoded from the information packets. As the definition of sub-optimal erasure codes, fountain codes are designed to allow the recovery of the original  $k$  information packets from any  $k$  transmission packets, where  $k$  is slightly larger than  $k$ .

The first practical realisation of a fountain code is the *Luby Transform* (LT) code [38]. LT codes have a similar graphical representation to LDPC codes. However, its graphical representation is implicit rather than explicit, in the sense that the current encoding process can be only tracked by its previous process and there is no predetermined global view of the graph. The header section of each transmission packet must contain the information on the connections between the information and parity-checks and its degree function. Most of time, the information in the header section is represented by a generation seed. The sender and receiver must be pre-synchronised and both agree with the applied degree distribution function and the generation seed. Luby [38] proved that by employing the encoding process with average degree  $O(\ln k)$ , LT codes can be devised to recover  $k$  information packets with  $k + \epsilon k$  encoded/transmission packets. The transmission packets are generated "on-the-fly" in term proportional to  $\ln k$ , and the recovery process (the XOR decoding process) requires only  $k \ln k$  computational time. However, LT codes suffer poor error-floors because of the latent *stopping-sets* in them.

## 1. INTRODUCTION

---

Raptor codes [68] improves the performance of LT codes by applying a pre-coding technique to circumvent the error-floor problem. Raptor codes currently give the best approximation to a digital fountain. The generation of a Raptor code can be viewed as a two-layered encoding process. The outer code is a fixed-rate erasure code, for example, LDPC codes or BCH codes, which aims to recover the unsolved erasures from the inner code. The inner code is an LT code, which maintains the desirable rateless property of a fountain code.

In 2002, a company called *Digital Fountain* was founded. The technology of fountain codes have been used for advanced IPTV, mobile broadcast, streaming video, file transfer and national defence applications, and are recognised by leading international standards bodies including DVB, 3GPP and IETF.

### 1.4 Basic Definitions

Consider a source that produces symbols from an alphabet  $\mathcal{A}$  having  $q$  symbols, where  $\mathcal{A}$  forms a field. We refer to a tuple  $(c_0, c_1, \dots, c_{n-1}) \in \mathcal{A}^n$  with  $n$  elements as an  $n$ -vector or an  $n$ -tuple.

**1.1 Definition (Block Code).** An  $(n, k)$  block code  $\mathcal{C}$  over an alphabet of  $q$  symbols is a set of  $q^k$   $n$ -vectors called **codewords** or **code vectors**. Associated with the code is an **encoder** which maps a **message**, a  $k$ -tuple  $\mathbf{m} \in \mathcal{A}^k$ , to its associated codeword.

For the purpose of the error correction/detection, one-to-one correspondence between a message  $\mathbf{m}$  and its codeword  $\mathbf{c}$  is required. However, for a given code  $\mathcal{C}$ , there exist more than one possible way of mapping messages to codewords.

A block code can be represented as an exhaustive list, but for large  $k$ , the storage memory and computational complexity may be prohibitively complex. The complexity can be reduced by imposing some sort of mathematical structure on the code. The most common requirement is *linearity*.

**1.2 Definition (Linear code).** Let  $\mathbb{F}_q^n$  denote an  $n$ -dimensional vector space over a finite field of  $q$  elements,  $\mathbb{F}_q$ . An  $[n, k, d]_q$  linear code  $\mathcal{C}$  is a  $k$ -dimensional subset



of  $\mathbb{F}_q^n$ . The quantity  $d$  is called the minimum Hamming distance of the code. Each vector in the  $k$ -dimensional subset of  $\mathbb{F}_q^n$ , which has length of  $n$  symbols, is called a codeword and may be denoted as  $\mathbf{c} = (c_0, c_1, \dots, c_{n-1})$ .

The term *linear* arises from the fact that a codeword may be obtained from linear combinations of other codewords and the component-wise sum of all codewords is an all-zero vector. It is assumed that operations such as addition and multiplication are performed under the algebra of  $\mathbb{F}_q$ .

**1.3 Definition (Code rate).** The code rate of an  $[n, k, d]_q$  linear code  $\mathcal{C}$  is the ratio  $k/n$ , denoted as  $\mathcal{R}$ .

**1.4 Definition (Hamming weight).** For a vector  $\mathbf{v} = (v_0, v_1, \dots, v_{n-1}) \in \mathbb{F}_q^n$ , the Hamming weight of  $\mathbf{v}$ —denoted by  $\text{wt}_H(\mathbf{v})$ , is the number of non zero elements in the vector. That is

$$\text{wt}_H(\mathbf{v}) = |\{v_i \neq 0 \mid 0 \leq i \leq n-1\}|.$$

**1.5 Definition (Systematic).** Let  $C$  be an  $(n, k)$  block code (not necessarily linear). An encoder is *systematic* if the message symbols  $m_0, m_1, \dots, m_{k-1}$  may be found explicitly and unchanged in the codeword. That is, there are coordinates  $i_0, i_1, \dots, i_{k-1}$  (which are most frequently sequential,  $i_0, i_0 + 1, \dots, i_0 + k - 1$ ) such that  $c_{i_0} = m_0, c_{i_1} = m_1, \dots, c_{i_{k-1}} = m_{k-1}$ . For a linear code, the generator for a systematic encoder is called a *systematic generator*.

**1.6 Definition (Hamming Distance).** The *Hamming Distance* between a sequence  $X = \{x_0, x_1, \dots, x_{n-1}\}$  and a sequence  $Y = \{y_0, y_1, \dots, y_{n-1}\}$  is the number of positions that the corresponding elements differ:

$$d(X, Y) = \sum_{i=0}^{n-1} (x_i \neq y_i) \quad (1.5)$$

## 1. INTRODUCTION

---

**1.7 Definition (minimum Hamming distance).** The *minimum Hamming distance* of a code is the smallest Hamming distance between any two codewords in the code.

**1.8 Definition (Field).** A *field*, denoted as  $\mathbb{F}$ , is a set of elements in which it is possible to add, subtract, multiply and divide (except that division by 0 or is not defined).

**1.9 Definition (Galois Field).** A *Galois Field*, also called as *finite field*, contains a finite number of elements, this number being called the *order* of the field, written as  $GF(x)$ , where  $x$  is the number of the elements.

## 1.5 Thesis Contributions and Outline

### 1.5.1 Contributions

Network transmissions require the applied erasure coding technique:

- with a strong erasure-recovery capacity
- with a reasonable computational complexity

These requirements are connected with the properties of erasure codes and the decoding algorithms. The research in this thesis was started from the study of the erasure decoding algorithms, which included the performance limits and the computational complexity of erasure decoding algorithms. Our first target was to design an erasure decoding algorithm to achieve an Maximum Likelihood (ML) performance with a reasonably computational complexity.

The recently popular codes, LDPC codes and Turbo codes, were firstly applied as the erasure codes. The common decoding algorithm for LDPC codes and Turbo codes is the *belief propagation* (BP) algorithm, which has demonstrated empirically near-optimal performance in the AWGN channel. In the application for the BEC, the BP decoding algorithm is also called as the *recovery* algorithm, which is to pass message through the bipartite graph of the code. Instead of

applying the recovery algorithm through the bipartite graph, we devised a matrix representation called the *erasure matrix* which only contains the information related to the erasures. Then, the stopping sets are explicitly expressed in the corresponding erasure matrix.

The performance of the recovery algorithm frequently suffer in the error-floor problem, which may be caused by the *stopping-sets* [16] due to the code structure.

Inspired by the bit-flipping BP algorithm in the Binary Symmetric Channel (BSC) [24], we designed the *guess* and *multi-guess* algorithms based on the recovery algorithm to break the stopping sets under the restriction of the maximum number of guesses. However, the computational complexity of the guess/multi-guess algorithms exponentially increases on the basis of the number of guesses which have been proceeded. The conventional ML erasure decoding algorithm has been known as the Gaussian Elimination algorithm which is literally to solve the parity-check equations at receiver. The Gaussian Elimination algorithm deploys a lot of column permutations and the row additions. This motivated us to study the relationship between the erasures and the parity-check polynomials in the parity-check matrix. We designed and analysed a novel ML algorithm – the In-place algorithm for decoding erasure codes in the BEC.

The invention of the complexity-reduced In-place algorithm diverged the research into two directions.

1. the application of the In-place algorithm with the soft-decision received data for the popular AWGN channel
2. to design an efficient transmission/packetisation structure to further reduce the computational complexity to approach a linear complexity

Re-order the received sequence with the values of the channel reliabilities from the less to the more or from the more to the less. The bits with the less reliabilities can be viewed as the erasures and those with the more reliabilities can be considered as the non-erasures. Henceforth, the In-place algorithm is equivalent to the ordered-statistic decoding (OSD) algorithm with the order of zero. The OSD algorithm is a well-known ML-approachable algorithm. However, the number of orders to achieve an ML performance is rarely known for a given

## 1. INTRODUCTION

---

code. This motivated us to search an alternative algorithm which would be capable of achieving an ML performance without pre-setting the number of orders. More importantly, this algorithm is preferable to be a flexible structure to avoid unnecessarily computational operations. Based on the tree structure of a code, we devised a branch-evaluation-search-on-the-code-tree (BESCT) algorithm.

The second direction was set to designate an efficient transmission/packetisation structure to further reduce the computational complexity to approach a linear complexity. This motivated us to study erasure codes for the network transmissions with packets. The performance limits of fountain codes showed that only under certain conditions of the length of fountain codes, fountain codes can perform sub-optimally with the actual number of the required packets close to the number of the information packets. To achieve the ML performance, it was natural to think of RS codes because they are optimal erasure codes. We designed and analysed the product packetisation structure with the in-place algorithm for the RS code transmission. The new arrangement performs much better than previously known algorithms, especially for fountain codes, and consumes less encoding and decoding times (as well as the computational complexity) than the conventional algorithms for RS codes. We believe that such decoding structure can also dramatically improve the performance of RS codes in the Rayleigh fading channel. Due to the existence of errors in the Rayleigh fading channel, we devised a concatenated BCH code with a product-packetised RS code to decode the errors and recover the erasures by a hard-decision (HD) algorithm.

### 1.5.2 Thesis Outline

In this section, we give a more detailed outline of the contents and contributions of this thesis. The thesis is partitioned into two parts as the reasons mentioned in previous section. Each part or each chapter can be read separately.

*Chapter 2: Matrix-based Erasure Decoding Algorithms [10, 74, 76]*

We present the erasure decoding algorithms on the matrix representation of erasure codes. The proposed erasure matrix is derived from the parity check matrix of the applied code which only contains the parity check polynomials with

erasures. The iterative recovery algorithm for LDPC codes exhibits the undesirable error-floors due to the stopping sets caused by the code structures. Inspired by the bit-flipping algorithm in the BSC, we devise the guess/multi-guess algorithm based on the recovery algorithm. In this chapter, we also introduce the method to make effective and efficient guesses and compare the simulation results. It is shown that the performances of the crucial guess algorithm are better than those of the random guess algorithm with different levels. The level of the improvement is determined by the code structure. To achieve an ML performance with a reasonable decoding complexity, we devise an In-place algorithm which is derived based on the Gaussian Elimination algorithm. Instead of process the whole parity-check matrix, the In-place algorithm only processes  $\epsilon$  parity-check polynomials in which erasures appear. Moreover, the In-place algorithm does not require any column-swapping process as the Gaussian Elimination algorithm which also reduce the computational complexity. In Section 2.5, a discussion on the relationship between the number of recoverable erasures and the weight distribution of the applied code is given.

### *Chapter 3: Branch-Evaluation Search on the Code-Tree Algorithm [9]*

Motivated by the invention of the In-place algorithm, we study the ML-approaching decoding algorithms in the AWGN channel to solve the errors when the soft-decision data have been received. Following a snapshot of the ordered-statistic decoding algorithm, we introduce the code tree representation of a linear code. Different to the conventional tree representation of a code, a bi-directional code tree is proposed in Section 3.3. By partitioning the code tree at a given cut-point vertex, the code tree grows potentially  $2^k$  branches. Each branch represents a valid codeword. Inspired by the Dorsch algorithm and the OSD algorithm, we propose the Branch-Evaluation-Search-on-the-Code-Tree (BESCT) algorithm which utilises the hierarchical property of the tree structure combined with the correlation metrics to realise an ML decoding (MLD).

### *Chapter 4: The Packet Data Transmission System of Fountain Codes [10]*

The focus of Chapter 4 is on the construction of fountain codes. The first realisation of fountain codes, as known as LT codes, is first studied. Investiga-

## 1. INTRODUCTION

---

tion of the degree distribution function and the asymptotic performance using the required packet counter reveals that only when the information file is large enough, the sub-optimal performance can be achieved. Then, we further investigate the upgraded fountain code, the Raptor code. Even though the outer code is deployed to solve the error-floor problem of LT codes, when the information file is not large enough, the performance of Raptor codes is hard to achieve the sub-optimal performance.

*Chapter 5: Capacity Approaching Codes for the Binary Erasure Channel Using a Product Packetisation Method [11, 74]*

RS codes, known as a class of maximum distance separable (MDS) codes, have amazingly attractive properties which make them as one class of optimal erasure codes. The drawback of RS codes in practice, comes from their non-binary structures which have been recognised as the reasons to their high encoding/decoding complexity. We introduce a product packetisation structure for all linear codes, in particular for non-binary codes, such as RS codes. The structure is named after the way of pre-storing the transmission data visually. Imagine there is a square or rectangular matrix, where the encoded data sequences are stored vertically and then the transmission packets pick up the data horizontally. In Section 5.2, the product arrangement matrix is designed to specify the proposed packetisation/de-packetisation structure. With the packet information in the header of each transmission packet, RS codes are capable of realising a rateless transmission which is always desirable for network transmissions. Figure 5.2 in Section 5.2 illustrates how the protocol of rateless RS coding scheme works. By deploying our proposed In-place algorithm at receiver, rateless RS codes can achieve their ML performance.

*Chapter 6: Concatenated Reed-Solomon Coding with Hard-decision for the Rayleigh Fading Channel [12]*

As the characteristics of the Rayleigh fading channel, the existence of additive errors and burst erasures requires the applied coding scheme with strong error and erasure correct-abilities. We present a hard-decision In-place algorithm scheme for concatenated RS codes offering both complexity and performance advantages over

## 1.5 Thesis Contributions and Outline

---

iterative decoding algorithms for optimally designed LDPC codes. The proposed RS coding scheme is to concatenate a Hamming or BCH code to detect and correct multiple hard decision errors in each received packet. Section 6.3 describes the system arrangement in details. The analysis of concatenated RS codes is given in Section 6.4, which also derives the performance bound of concatenated RS codes.

The reader should be aware that some notations may be used repetitively in this thesis. Please refer them to the definitions/designations in the chapters where they appear.

## 1. INTRODUCTION

---



## Part I

# Erasure Decoding Algorithms

*The road to success is always under construction.*

Lily Tomlin

# 2

## Matrix-based Erasure Decoding Algorithms

### 2.1 Background

The binary erasure channel (BEC) is a well-known model for the Internet transmission. This channel was introduced by Elias [19] in 1955. With a packet lost due to network congestion with probability of  $p$ , the BEC has a capacity of  $1 - p$ . Elias proved that there exist codes of rate  $R$  for any  $R < 1 - p$  that can be used to transmit over channels of capacity  $1 - p$ .

Although maximum likelihood decoding (MLD) of linear block codes is known to be NP-hard [5], *iterative decoding*, as a technique deploying message-passing algorithm recursively to improve the code performance, aims to approach an MLD.

The history of iterative decoding algorithms can be tracked back to 1954 when Elias [18] published his work on *iterated codes*. Ten years after that, in

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

the 1960s, Gallager [24] and Massey [44] made important contributions and then iterative decoding was referred to as *probabilistic decoding*. The target of iterative decoding is to maximise the *a-posterior probability* of a bit/symbol being sent given on noisy version of the coded sequence. Iterative Belief Propagation (BP) decoding algorithm was proposed by Pearl [51].

One of the main motivations behind this chapter is to devise ML or ML-approachable decoding algorithms for the BEC.

We start the work with the review of the recovery algorithm to correct the erasures for the BEC, in which each codeword bit is lost with a fixed constant probability  $p$  in transit independent of all the other bits in Section 2.2. Instead of deploying the bipartite graph, we introduce matrix representations of the erased bits by superposition of the erased bits on the parity-check matrix for iterative decoding algorithms over the BEC. Following a quick re-format of the recovery algorithm in matrix representations (Section 2.3), we combine the idea of Gallager's bit-flipping algorithm and the recovery algorithm into the guess/multi-guess algorithm in particular for LDPC codes in Section 2.3.3. We then attempt to reach our target to design an ML decoding algorithm for the BEC. In Section 2.4, inspired by the ML Gaussian Elimination (GE) algorithm, we propose a novel, non-iterative ML-approachable decoding algorithm – the In-place algorithm which is with a reduced complexity in comparison to the GE algorithm. We provide the numerical results and the corresponding discussions in Section 2.5, where we also analyse the relationship between the marginal number of correctable erasures and the average number of correctable erasures. In Section 2.6, we conclude this chapter and highlight its main results.

### 2.2 A Review of the Recovery algorithm

Luby *et al.* [39] simplified the BP decoding algorithm in conjunction with the characteristics of the BEC. The so-called “recovery” algorithm can be visualised by a bipartite graph between variable vertices and the check vertices as shown in Figure 2.1.

Putatively from the bipartite graph shown in Figure 2.1, there are two sets of vertices required for an implementation of the Recovery algorithm, including

## 2.2 A Review of the Recovery algorithm

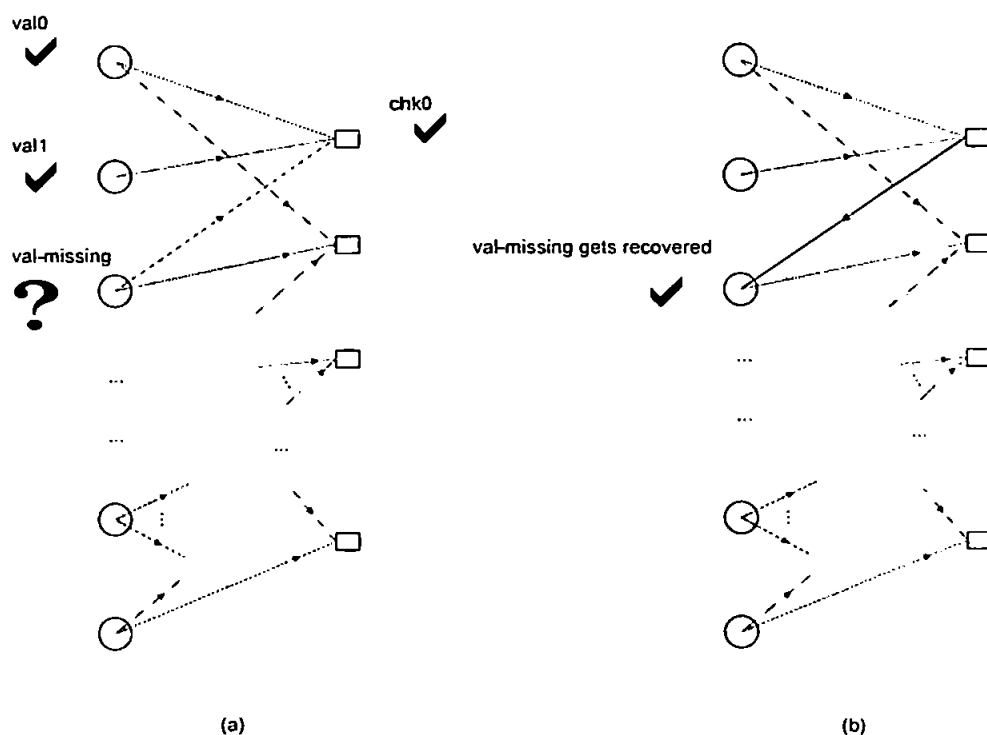


Figure 2.1: the Recovery Algorithm over the Bipartite Graph: square spots representing the check vertex and circle spots denoting the variable vertex

the variable set  $val$  and the check set  $chk$ . Denote the missing symbol labelled by a "?" as  $val_{missing}$  which is connected to the first check vertex, written as  $chk_0$ , on its right-hand side and further related to other two variable vertices via the connected check vertex. Both related variable vertices are just adjacent to  $val_{missing}$ , noted as  $val_0$  and  $val_1$  respectively. Then, the value of this missing symbol is implicitly calculated by the connected check vertex and its related adjacent variable vertices as  $val_{missing} = chk_0 \oplus val_0 \oplus val_1$ . Then by exclusive-oring the value of  $val_{missing}$ , all other connected vertex are updated. Till now, we call it as one *recovery* process. By recursively implementing the recovery process, the recovery decoder can decode the received erasure successfully or failed due to the existence of stopping sets [16].

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

**2.1 Definition (Stopping Set).** A *stopping set*, denoted as  $\mathcal{S}$ , is the set of variable vertex such that all neighbours of  $\mathcal{S}$  are connected to  $\mathcal{S}$  for at least twice.

Example 2.1 illustrates a stopping set example with a bipartite graph.

---

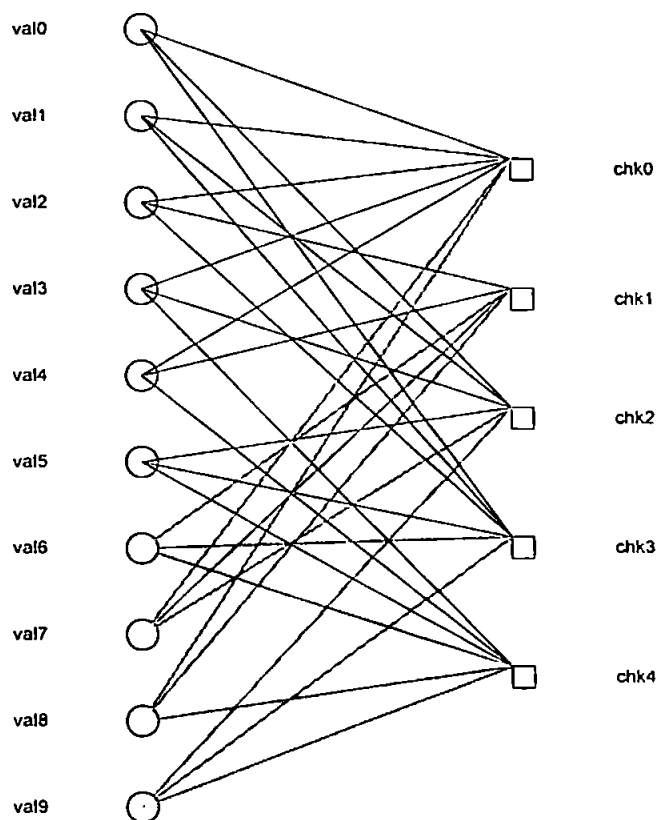


Figure 2.2: an Example of a Bipartite Expression of a Code

## 2.2 A Review of the Recovery algorithm

---

**Example 2.1:** Figure 2.2 gives a graphical expression of a LDPC code with its ensemble of  $\mathcal{C}(10, x^3, x^6)$ . For the variable vertex indexed from  $val_0$  to  $val_9$ , the connection relationships are listed as:

$$\begin{aligned}
 val_0(chk_0, chk_2, chk_3) & \quad val_1(chk_1, chk_2, chk_3) \\
 val_2(chk_0, chk_1, chk_3) & \quad val_3(chk_0, chk_2, chk_4) \\
 val_4(chk_0, chk_1, chk_4) & \quad val_5(chk_2, chk_3, chk_4) \\
 val_6(chk_1, chk_3, chk_4) & \quad val_7(chk_0, chk_1, chk_2) \\
 val_8(chk_0, chk_1, chk_4) & \quad val_9(chk_2, chk_3, chk_4).
 \end{aligned}$$

Group the variable vertices of  $val_6$ ,  $val_7$ ,  $val_8$  and  $val_9$  as a set  $\mathcal{S}(6, 7, 8, 9)$ . It is noticeable that all the check vertices are connected with  $\mathcal{S}(6, 7, 8, 9)$  at least twice. Alternatively, we can also determine the stopping sets by observing the parity-check matrix  $H$  of the code. As expressed below, the identity part at the RHS of the  $H$  indicates the check vertices and all the 1s in the LHS of the  $H$  represent the connections between the check vertices and the variable vertices.

$$H = \left( \begin{array}{cccc|cccc}
 1 & 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
 1 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1
 \end{array} \right) \quad (2.1)$$

Clearly, each parity-check equation contains more than one variables from the stopping set  $\mathcal{S}(6, 7, 8, 9)$  as highlighted, which also further implies that if the symbols in  $\mathcal{S}$  are all lost after a transmission, it can not be recovered from any parity-check equation in  $H$ .

---

\*an ensemble of a LDPC code [60] written as  $\mathcal{C}(n, \lambda, \rho)$  is defined by the code length of  $n$ , a variable vertex degree distribution  $\lambda$  and a check vertex degree distribution  $\rho$

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

### 2.3 Matrix-based Iterative Decoding Algorithm and its Variances via the BEC

Instead of deploying the bipartite graph to represent the code, we use the parity-check matrix  $\mathbf{H}$  to express the code.

$$\mathbf{H} := \begin{pmatrix} h_0 \\ h_1 \\ \dots \\ h_{n-k-1} \end{pmatrix}$$

in which each row  $h_i$  can be written as a parity-check equation as  $h_i = h_{i,0}x^0 + h_{i,1}x^1 + \dots + h_{i,n-1}x^{n-1}$ .

Considering an  $(n, k)$  binary linear block code, we denote a codeword as  $\mathbf{x} = \{x_0, x_1, \dots, x_{n-1}\}$ . After being transmitted over the BEC with the erasure probability of  $\epsilon$ . The received vector, designated as  $\mathbf{y}$  contains two parts: the transmitted sub-sequence  $\mathbf{y}_{rec}$  and the erased sub-sequence  $\mathbf{y}_\epsilon$ ,

$$\mathbf{y}_{rec} = \{y_0^r, y_1^r, \dots, y_{l-1}^r\} \quad (2.2)$$

$$\mathbf{y}_\epsilon = \{y_0^\epsilon, y_1^\epsilon, \dots, y_{z-1}^\epsilon\} \quad (2.3)$$

where  $l + z = n$ . We also define a vector  $\mathbf{f} = \{f_0, f_1, \dots, f_{n-1}\}$  which maps the erased bits as "1"s and the received bits as "0"s.

#### 2.3.1 Matrix Representation of the Erased Bits

A matrix representation of the erased bits, designated as  $\mathbf{M}$ , is developed from the parity-check matrix  $\mathbf{H}$ . By the reflection of the position of each erasure in  $\mathbf{H}$ , the proposed *erasure matrix* with its size of  $z \times n$ , denoted as  $\mathbf{M}$ , is defined

### 2.3 Matrix-based Iterative Decoding Algorithm and its Variances via the BEC

---

as in (2.4)

$$M := \begin{pmatrix} \mathcal{M}_0 \\ \mathcal{M}_1 \\ \dots \\ \mathcal{M}_{z-1} \end{pmatrix}. \quad (2.4)$$

The component equations in  $M$  can be derived as follows:

$$\begin{aligned} \mathcal{M}_0 &= (h_{0,0} \cdot f_0) + (h_{0,1} \cdot f_1)x^1 + (h_{0,2} \cdot f_2)x^2 + \dots + (h_{0,n-1} \cdot f_{n-1})x^{n-1} \\ \mathcal{M}_1 &= (h_{1,0} \cdot f_0) + (h_{1,1} \cdot f_1)x^1 + (h_{1,2} \cdot f_2)x^2 + \dots + (h_{1,n-1} \cdot f_{n-1})x^{n-1} \\ &\vdots \\ \mathcal{M}_{z-1} &= (h_{z-1,0} \cdot f_0) + (h_{z-1,1} \cdot f_1)x^1 + (h_{z-1,2} \cdot f_2)x^2 + \dots + (h_{z-1,n-1} \cdot f_{n-1})x^{n-1} \end{aligned}$$

where  $z \leq n - k$ .<sup>†</sup>

Moreover, two sets of variables are devised to identify the erasure positions in  $M$  from different observation angles. The index  $i$  indicates the elements in each row with the range of  $[0, \dots, n - 1]$  and another index  $j$  indicates the elements in each column with the range of  $[0, \dots, z - 1]$ . Then, the notation of  $\mathcal{M}_j(i)$  represents the element contained in the  $j^{\text{th}}$  component equation and located at the  $i^{\text{th}}$  position.

$$\begin{aligned} E^h &:= \{E_0^h, E_1^h, \dots, E_{z-1}^h\} \\ E^v &:= \{E_0^v, E_1^v, \dots, E_{n-1}^v\} \end{aligned} \quad (2.5)$$

In (2.5),  $E^h$  and  $E^v$  are designated as the sets of erased bits participating in each row and column of  $M$ , where the superscripts  $h$  and  $v$  represent ‘‘horizontal’’ and ‘‘vertical’’ respectively. Also, for convenience,  $|E_i^h|$  is called the profile of  $i^{\text{th}}$  horizontal component which is the number of erased bits in  $E_i^h$  and  $|E_j^v|$  is called

---

<sup>†</sup> $z = n - k$  when the result of each parity-check equation mapping the vector  $f$  is counted more than 1.



## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

the profile of  $j^{\text{th}}$  vertical component which is the number of erased bits in  $E_j^v$ .

**Example 2.2:** Based on the  $H$  from Example 2.1, if the variables in  $\mathcal{S}$  are all erased after a transmission, the vector  $\mathbf{f}$  will be (00000011110000) and the  $M$  will be mapped out from the parity-check matrix  $H$  (2.1) as :

$$M = \begin{matrix} & \text{index} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} & & \left( \begin{array}{cccccccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{array} \right) \end{matrix}$$

The corresponding  $E$  vectors are:

$E^h$	$E^v$
$E_0^h = (7, 8)$	$E_6^v = (1, 3, 4)$
$E_1^h = (6, 7, 8)$	$E_7^v = (0, 1, 2)$
$E_2^h = (7, 9)$	$E_8^v = (0, 1, 4)$
$E_3^h = (6, 9)$	$E_9^v = (2, 3, 4)$
$E_4^h = (6, 8, 9)$	

And therefore, both the profiles for the horizontal component and the vertical component are  $|E^h| = \{2, 3, 2, 2, 3\}$  and  $|E^v| = \{0, 0, 0, 0, 0, 0, 3, 3, 3, 3, 0, 0, 0, 0, 0\}$  respectively.

---

### 2.3.2 Matrix-based Iterative Recovery Algorithm

After giving the concept of an erasure matrix, the standard graph-based iterative decoding algorithm can also be represented as a matrix-based iterative recovery algorithm.

By defining an erasure matrix  $M$  based on the received vector  $\mathbf{y}$  and its parity-check matrix  $H$ , the matrix-based recovery algorithm starts with an evaluation

### 2.3 Matrix-based Iterative Decoding Algorithm and its Variances via the BEC

---

on the horizontal profile on each row of  $M$ . For the  $i^{\text{th}}$  row  $\mathcal{M}_i$ , if  $\exists |E_i^h| = 1$ ,

$$y_{E_i^h}^c = \sum_{j=\{0,1,\dots,n-1\}\setminus E_i^h} \oplus y_j \cdot h_{i,j} \quad (2.6)$$

Then remove  $y_{E_i^h}^c$  from the erasure set  $\mathbf{y}^c$  and meanwhile update the matrix  $M$ . Repeat the evaluation process and the recovery process until all the erased bits being solved, or the decoding cannot continue further, that is

$$\forall i \in \{0, 1, \dots, z-1\}$$

$$|E_i^h| = \begin{cases} 0, & \text{no more erasures} \\ \delta, & \delta \geq 2 \end{cases} \quad (2.7)$$

In (2.7), the existence of  $\delta$  implies a decoding failure caused by the stopping sets. It has been proved in [16] that the recovery algorithm fails if and only if  $\mathbf{y}^c$  contains some stopping sets. Di *et al.* [16] also has proved that the set of the remaining erasures, when the iterative decoding algorithm fails, is same as the unique maximal stopping sets of  $\mathbf{y}^c$ . The algorithm is given as follows as *Algorithm 2.1*.

---

#### Algorithm 2.1 RecoveryDec( $n, \mathbf{H}, \mathbf{y}$ )

---

**Input:**

- $n \leftarrow$  code length (block length)
- $\mathbf{H} \leftarrow$  its parity-check matrix
- $\mathbf{y} \leftarrow$  received vector containing erasures

- 1: **repeat**
  - 2:   **for all**  $h_i \in \mathbf{H}$  : each parity-check polynomial **do**
  - 3:     **if**  $(\text{Num}(h_i)_c == 1)$  **then**
  - 4:        $y_c = \sum_{j=\{0,1,2,\dots,n-1\}\setminus j \setminus \text{pos}(y_c)} \oplus h_{i,j} \cdot y_j$
  - 5:        $\text{Num}(c) \leftarrow$  --
  - 6:        $\text{Set}_c = \text{Set}_c \cup y_c$
  - 7:     **end if**
  - 8:   **end for**
  - 9: **until**  $\text{Num}(c) = 0$  or  $\text{Num}(c) < 2$
-

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

There are several variables and subroutines in *Algorithm 2.1* are listed as following.

- $M$  erasure matrix generated by  $H$
- $Num(\cdot)$ , returns the values of  $|E^h|$
- $Set_i$  contains the set of  $y^e$
- $pos(\cdot)$  returns the position of the erasure regarding to  $H$
- $i \Leftarrow$  an integer used as an index

The schematic diagram of the recovery algorithm is also illustrated in Figure 2.3. The iterative solution sequencer works as the evaluation process on the erasure matrix  $M$ , and the recovery box solves all the parity check equations with single bit erasure.

Considering the nature of the recovery algorithm, the decoding complexity is straightforward related to the number of erasures in the received vector, which can be estimated by the code length  $n$  and the average erasure probability  $p$  of the BEC.

If there are  $\varepsilon$  erasures in  $\mathbf{y}$ , the decoder starts to solve the parity-check equation with a single erasure, where  $\varepsilon = \varepsilon n$  and  $\varepsilon$  is an erasure probability of the BEC. As an instance, there is a single erasure in the  $i^{th}$  equation  $\mathbf{h}_i$ , as depicted in (2.6), the single recovery process involves  $wt(\mathbf{h}_i)$ <sup>†</sup> multiplications and  $wt(\mathbf{h}_i) - 1$  exclusive-or operations. Clearly, both the multiplication and the exclusive-or operation are determined by the weights of the erased parity-check equations. If the applied code is a regular LDPC code with its ensemble of  $\mathcal{C}(n, \lambda, \rho)$  and  $\begin{pmatrix} \lambda \\ \rho \end{pmatrix} = \begin{pmatrix} x^a \\ x^b \end{pmatrix}$ , the average number of multiplications is  $nc \cdot (b + 1)$  ( $= wt(\mathbf{f}) \cdot (b + 1)$ ) and its average number of exclusive-or operations is  $nc \cdot b$  ( $= wt(\mathbf{f}) \cdot b$ ).

---

<sup>†</sup> $wt(\cdot)$  returns the weight of a vector.

## 2.3 Matrix-based Iterative Decoding Algorithm and its Variances via the BEC

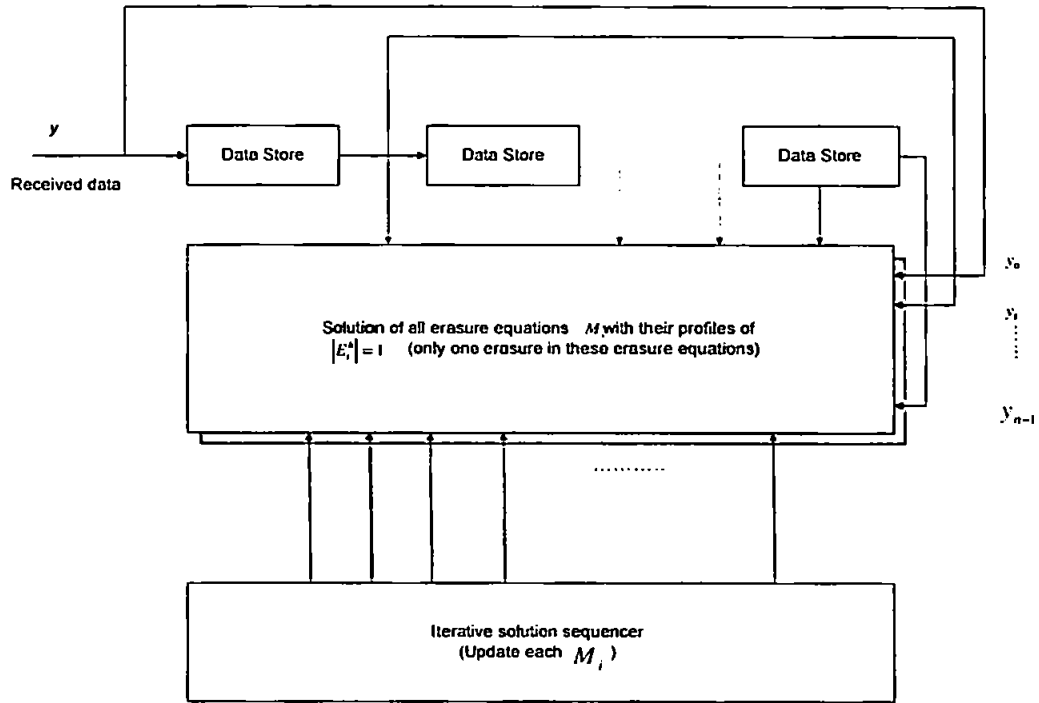


Figure 2.3: Matrix-based Iterative Recovery Decoder

Generally speaking, for a successful decoding process, the decoder complexity is linearly towards the  $\epsilon \cdot n$ , and henceforth the overall decoding complexity (or *decoding time*) can be derived as  $O(\epsilon \cdot n)$ <sup>§</sup>.

Figure 2.4 exhibits the recovery decoder performance via its *Bit Error Rate* (BER) and also the *Frame Error Rate* (FER) versus the erasure probability  $p$ . The applied code is the *Progressive Edge-Growth* [32] (PEG) LDPC code (256, 128) which achieves the performance of  $10^{-7}$  for the BER and less than  $10^{-5}$  for the FER at  $p = 0.25$ .

<sup>§</sup>*big Oh notation:*  $O(n)$  describes an algorithm whose performance will grow linearly and in direct proportion to the size of the input data set.

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

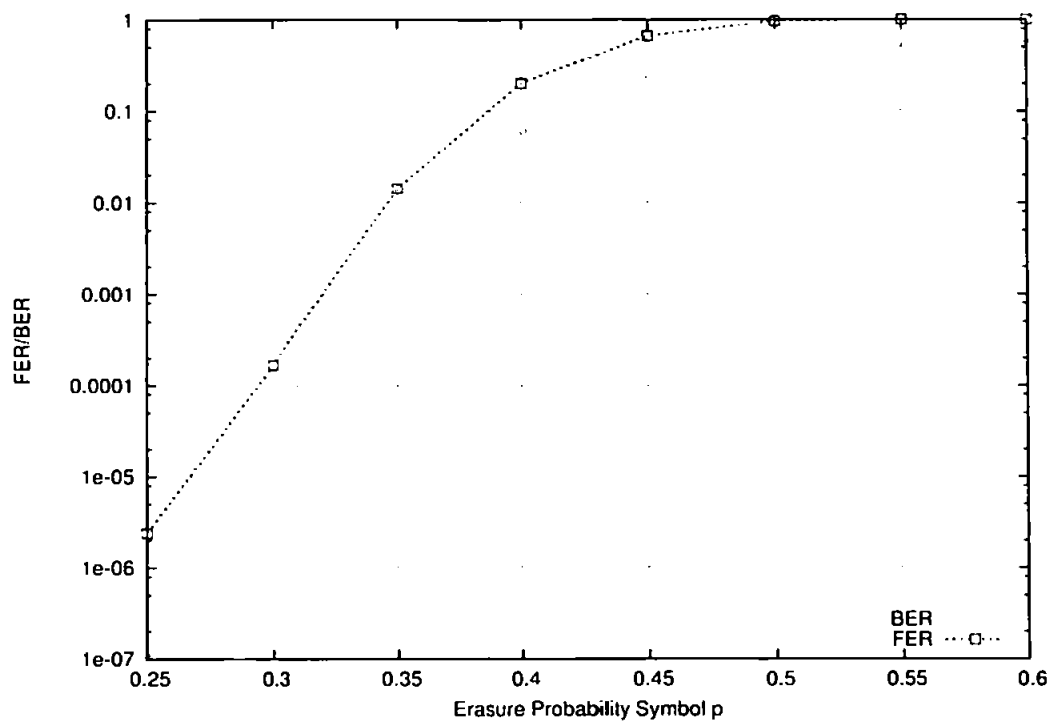


Figure 2.1: The BER/FER performance of the PEG LDPC code (256, 128) with the Matrix-based Recovery Algorithm

### 2.3.3 Guess/Multi-Guess Algorithm

As described in previous section, the matrix-based iterative recovery algorithm fails if and only if the erasures contain a stopping set  $\mathcal{S}$ .

To break the stopping sets which impedes the recovery algorithm, inspired by the idea of Gallager's bit-flipping algorithm, the *guess algorithm* [16] was first mentioned in 2002. In our work [10], we have a further discussion on this algorithm and extend it into the multi-guess algorithm.

Literally, the guess algorithm means to make assumptions on the left  $z'$  erasures when the recovery algorithm fails, where  $z' \leq z$ . Mapping the unsolved received vector to a new flag vector  $\mathbf{f}' = \{f'_0, f'_1, \dots, f'_{n-1}\}$ , the flag vector contains 2 or more than 2 elements equal to "1". The aim of the guess algorithm is to break the stopping sets and complete the decoding by a restricted number of "guesses".

The notation of  $g$  is employed to represent "guess" in this section. We also designate  $\mathbf{y}'_c$  as the left erasures and  $\mathbf{y}'_c \subseteq \mathbf{y}_c$ . When the recovery algorithm fails due to the stopping sets which cause the horizontal profile of  $\mathbf{M}$  is 2 or more than 2, the *guess algorithm* is triggered to continue the decoding as following. Choose one unsolved erasure  $y'_c(v_i)$  and then assign it as "1" or "0" for a binary transmission. For example, we assign "1" to  $y'_c(v_0)$  as the first guess erasure value and map  $f'_{v_0}$  as "0". Update the erasure matrix  $\mathbf{M}$  with the new  $\mathbf{f}'$  and therefore the mark at location  $v_0$  in each erasure equation  $\mathcal{M}_i$  is removed. Repeat the recovery process, which is to replace the single erasure in  $\mathcal{M}_i$  by the exclusive-or-ed result as depicted in (2.6). Suppose the recovery algorithm can successfully solve all the remaining erasures after a single guess  $g_0$ . Store the decoded sequence as  $c_{out_0}$  and then trace back the process at the  $v_0$  position. Re-assign "0" to  $y'_c(v_0)$  and repeat the recovery algorithm. Hereafter another decoded sequence  $c_{out_1}$  is also listed as a potential codeword.

A schematic diagram of the decoder is illustrated in Figure 2.5. The modification is developed at the iterative solution sequencer with an additional guess function.

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

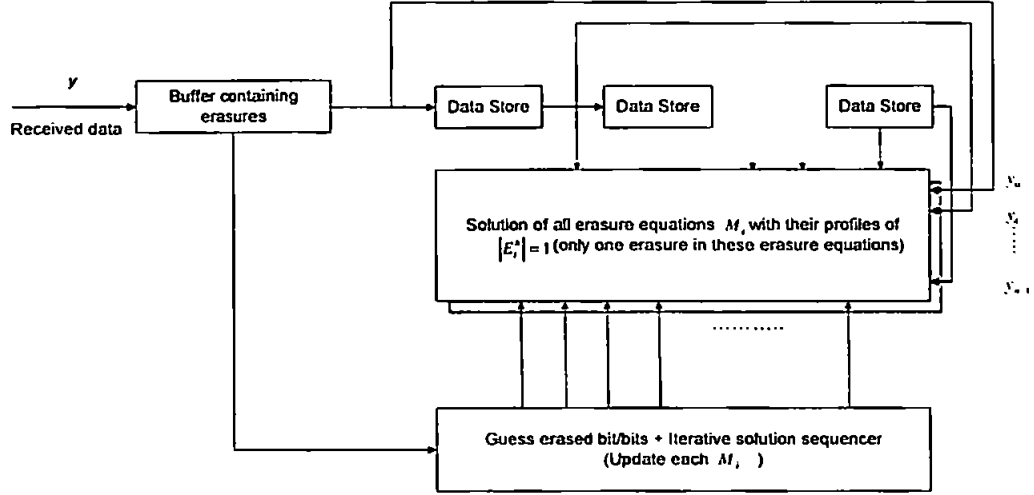


Figure 2.5: Correction of Erased Bits Using Guess Algorithm

Following the definition of a regular LDPC code, suppose the remaining erasures  $y'_e$  can be solved by  $g_0$  and the decoding complexity can be derived as

$$\begin{aligned} \text{multiplications:} & \quad [wt(\mathbf{f}) + wt(\mathbf{f}')] \cdot (b + 1) \\ \text{exclusive-or operations:} & \quad [wt(\mathbf{f}) + wt(\mathbf{f}')] \cdot b \end{aligned}$$

Therefore, from the linear relationship stated above, let  $T_{\text{Recovery}}$  be the decoding time for the Recovery algorithm, and  $T_{\text{Guess}}$  be that for the guess algorithm. The decoding times between the Recovery algorithm and the guess algorithm is also linear, which can be considered as

$$T_{\text{Guess}} = (1 + \eta) \cdot T_{\text{Recovery}} \quad (2.8)$$

where  $\eta \geq 0$ .

If the decoder after one guess stops again, another guess has to be made to

### 2.3 Matrix-based Iterative Decoding Algorithm and its Variances via the BEC

carry on the decoding process. Denote the number of guesses as  $ng$ . Obviously, the value of  $\eta$  is mainly determined by  $ng$ . If the code is a binary code, the number of potential codewords after  $ng$  guesses is  $2^{ng}$ . As an instance, assuming  $ng = 3$ , after the guess and recovery process, there will be 8 potential codewords for the final decision.

To compromise the code performance and the decoding complexity, we usually limit the number of guesses to a small number  $ng_{\max}$ . If after  $ng_{\max}$  guesses, the decoding still cannot be finished, a decoding failure is declared. For a low-rate LDPC code, the guess algorithm is able to improve the performance by equal to or less than 3 guesses. e.g. the performance of the PEG LDPC (256, 128) binary code in Figure 2.6. When  $p = 0.35$ , the performance of the guess decoder with  $ng_{\max} = 3$  has been improved by the magnitude order of 2 on the basis of that of the Recovery decoder in terms of the FER.

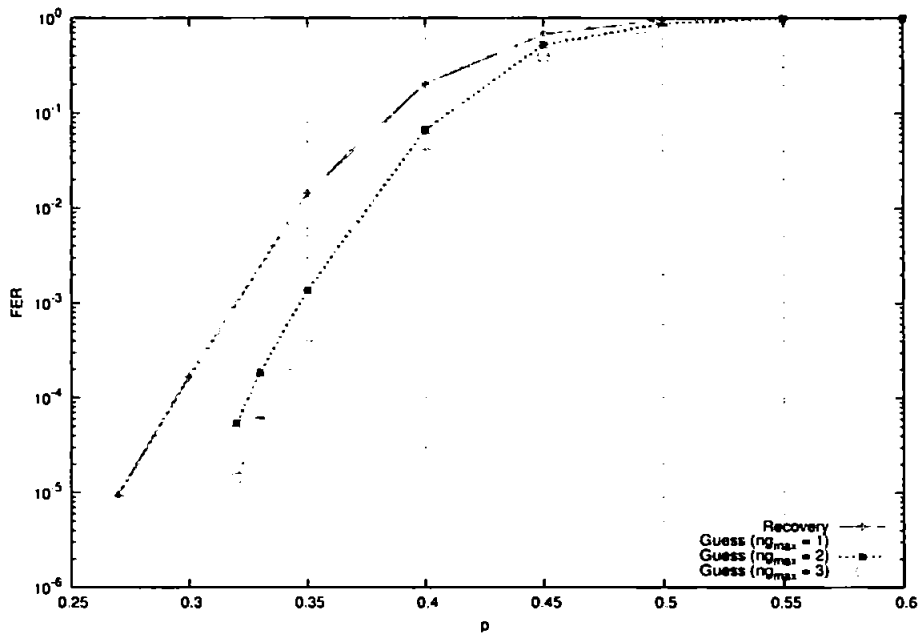


Figure 2.6: Performance of the PEG LDPC (256,128) with the Recovery Algorithm and C-Guess Algorithm

Another factor to affect the decoding performance and complexity is how to make guesses effectively and efficiently.



## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

- *sequential guess*: the guess bits are chosen based on the location of the unsolved erasures sequentially.
- *crucial guess*: the guess bits are chosen on the basis of the highest value of  $|E_j^c|$  with the value of  $|E_j^h| \geq 2$ .

Compared to the crucial guess algorithm, the sequential guess algorithm is straightforward but to perform worse than the crucial guess algorithm. For an irregular LDPC code <sup>¶</sup>, the performance gap between the crucial guess and the sequential guess is even more obvious than that for a regular LDPC code, as shown in Figure 2.7. More simulation results and discussions are given in Section 2.5.

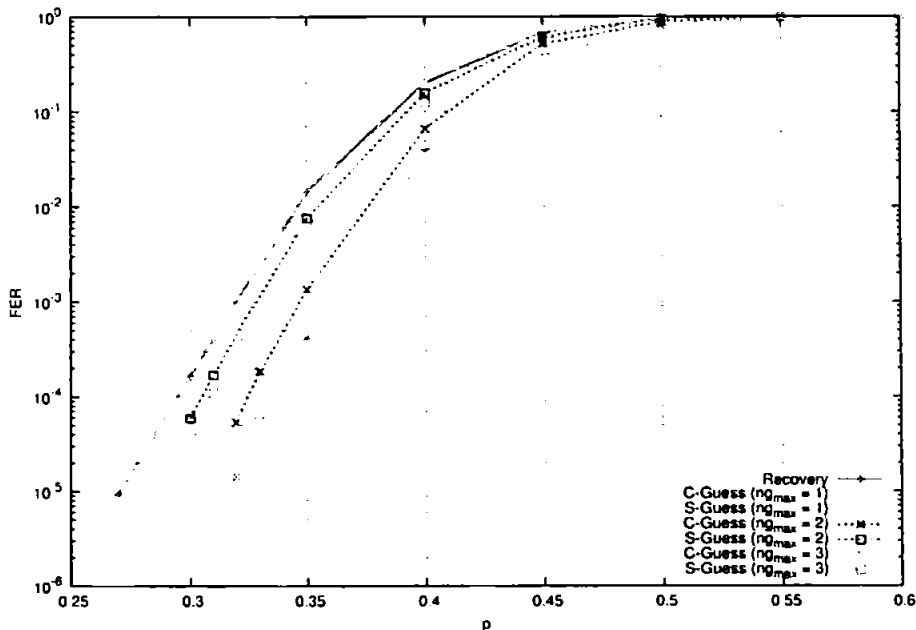


Figure 2.7: Performance Comparison between the Crucial Guess algorithm (C-Guess) and the Sequential Guess algorithm (S-Guess) of the PEG LDPC (256,128)

At the end of the decoding, we need to pick out one codeword from  $2^{n_g}$  potential codewords generated from the guess process as the decoded result. We call

<sup>¶</sup>An irregular LDPC code has its parity-check  $H$  with different column weights and row weights.

## 2.3 Matrix-based Iterative Decoding Algorithm and its Variances via the BEC

---

the final determination stage as the *evaluation stage*. From the list of potential codewords,  $c_{\text{out}_i}$ , where  $i \in \{1, 2, \dots, 2^{ng}\}$ , pick the one that satisfies  $\mathbf{H}c_{\text{out}_j}^T = \mathbf{0}$ . The evaluation stage obviously requires some operations which include  $2^{ng}$  matrix multiplications and  $2^{ng} - 1$  comparisons.

For sparse linear codes, a hybrid guess-increasing scheme works efficiently. However, for non-sparse linear codes, it is common to encounter more than 2 unsolved symbols in each erasure equation of  $\mathcal{M}$  after running the recovery Algorithm due to the high-density of their parity check matrix. In these cases, we cannot break the stopping set by guessing one erased bit/symbol in a row only. More than 1 erased symbols at one time need to be guessed. We can calculate the minimum number of guesses before the decoding.

**2.1 Lemma.** Consider the chosen erased bits/symbols in each row as an erased group. Let  $\omega_\delta$  denote the set of rows with  $\delta$  erasures, that is,  $\omega_\delta = \{i \mid |E_i^h| = \delta\}$ . And  $x_\delta$  is the set of rows which satisfies:

$$x_\delta = \{i \in \omega_\delta \mid \exists k, p \in E_i^h, \text{ such as } k \neq p, |E_k^v| = |E_p^v| = 1\}. \quad (2.9)$$

Then

$$\min(ng) = |x_\delta| + 1 \quad (2.10)$$

where 1 accounts for the need for at least one “crucial” row.

**Proof.** When the guessing process stops, there are more than 2 erased symbols in each erased row. The rows that have more than two bits  $(k, p)$  which do not participate in any other row (i. e.  $|E_k^v| = |E_p^v| = 1$ ) cannot be solved by other rows, and so at least one of these bits has to be guessed. So the minimum number of guesses equals to the number of all the independent guesses plus one more “crucial” guess to solve the other rows.  $\square$

For the Multi-Guess Algorithm, a whole row is guessed. A crucial row  $c$  is defined as follows:

1.  $c \in \omega_\delta$

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

2.  $\sum_{j \in E_c^h} |E_j^v|$  is maximised over  $c$  in  $\omega_\delta$

The one-stage multi-guess algorithm is given below:

### the One-stage Multi-Guess Algorithm

- *step 1* Run the decoder with Guess Algorithm until  $|E_i^h| \geq 2$  for  $i = 1, \dots, L_c$ .
- *step 2* Evaluate the value of  $\min(ng)$ . If  $\min(ng) > ng_{\max}$ , the decoding declares a failure and exits.
- *step 3* Group the rows with  $|E_i^h| = \delta$  as  $\omega_\delta$ , where  $i \in \{1, 2, \dots, L_c\}$ .
- *step 4* Find the “crucial” row and guess all erased bits in that row. (There will be at most  $2^{\delta-1}$  guesses.)
- *step 5* Guess one bit  $p$  with  $|E_p^v| = 1$  in each of the independent rows, i.e. the rows in  $x_g$ .
- *step 6* Update  $M_c$ ,  $\mathbf{E}^h$  and  $\mathbf{E}^v$ . Continue the decoding from step 3 to step 5 until all the erased bits are solved or the decoding cannot continue further.

The disadvantages of Guess and Multi-Guess Algorithms include the decoding complexity and the correctness of the results. The decoding complexity grows exponentially with the number of guesses. It is possible that the group guess declares a wrong value as the result of the decoder. Although this kind of situation happens only when the erasure probability is very small, it is still undesirable.

## 2.4 Matrix-based In-place Decoder

In this section, we devise a novel ML decoding algorithm to solve erasures in the BEC. The transmitted vector is defined as  $\mathbf{x} = \{x_0, x_1, \dots, x_{n-1}, \forall x_i \in \{0, 1\}\}$  and the received vector is written as  $\mathbf{y}$  which includes two parts: the uncrased part  $\mathbf{y}_{rec}$  and the erased part  $\mathbf{y}_\epsilon$ , that is,  $y_i \in \{0, 1, \epsilon\}$ <sup>||</sup>. We attempt to devise an algorithm to decode the erased bits by solving (2.11).

$$\mathbf{H}\mathbf{y}^T = \mathbf{0}. \quad (2.11)$$

Instead of using the erasure matrix  $\mathbf{M}$  which only contains the erasure positions on each parity-check equation, in this section, we split the parity-check matrix  $\mathbf{H}$  into  $\mathbf{H}_\epsilon$  and  $\mathbf{H}_{rec}$ . Then, Equation (2.12) can be derived as (2.12) as follows.

$$\mathbf{H}_\epsilon \mathbf{y}_\epsilon^T = \mathbf{H}_{rec} \mathbf{y}_{rec}^T \quad (2.12)$$

The right-hand side of Equation (2.12) can be calculated out as denoted as  $\mathbf{q}_{rec}^T$ . As long as an MLD is possible, the equation  $\mathbf{H}_\epsilon \mathbf{y}_\epsilon^T = \mathbf{q}_{rec}^T$  should have a unique solution, which is the case if and only if all the parity-check equations in  $\mathbf{H}_\epsilon$  are linearly independent [60]. The conventional Gaussian Elimination (GE) algorithm [48] requires a column-permutation process and an equation-solving process. Assume there are  $\epsilon$  erasures to be solved, that is, there are  $\epsilon$  equations to be solved in a linear system, and there are  $\varrho$  variables in each equation on average. The GE algorithm requires  $\varrho^2 \cdot \epsilon + \epsilon^3$  operations for a complete decoding. There are several fast methods proposed to solve a linear system of equations, such as the Strassen algorithm [71] which implements an iterative process on precalculated data and requires  $O(2^{2.81})$ , and Coppersmith algorithm [15] which requires  $O(2^{2.376})$  but very impractical for the purpose of decoding.

We now devise an improved ML approach – the *In-place algorithm*, which is inspired by the GE algorithm but with a reduced computational complexity.

This algorithm is divided into two parts: the *Polynomial-update Process* and the *Back-filling Process*.

---

<sup>||</sup> $i$  represents a valid index in vectors.

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

- Polynomial-update Process The codeword is received and  $y_i$  are substituted in positions of erased bits/symbols in  $H$ . Starting with one of the erased bits/symbols,  $y_{e_s}$ , the first equation containing this bit/symbol is flagged that it will be used for the solution of  $y_{e_s}$ . This equation is subtracted from all other equations containing  $y_{e_s}$  and not yet flagged to produce a new set of equations. The procedure repeats until either non flagged equations remain containing  $y_{e_s}$  (in which case a decoder failure is declared) or no erased bits/symbols remain that are not in flagged equations. The pseudo-code of the polynomial-update algorithm is given in *Algorithm 2.2*.

---

### Algorithm 2.2 Polynomial-update Algorithm

---

Input:

- $\mathbf{y} \Leftarrow$  received vector
  - $\mathbf{H} \Leftarrow$  original parity-check matrix of the code
  - 1: for all  $y_i^t \in \mathbf{y}$  do
  - 2:   flagging the first parity-check polynomial which containing  $y_i^t$  as  $h_i^t$
  - 3:   for all  $h_j \in \mathbf{H}$  all flagged polynomials do
  - 4:     if  $h_j(i) == 1$  then
  - 5:       push  $h_j$  into  $Set(\epsilon_i)$
  - 6:     end if
  - 7:     for all  $h_j \in Set(\epsilon_i)$  do
  - 8:        $h_j(k) = h_i(k) \oplus h_j(k)$ , for  $k = 0, 1, \dots, n-1$
  - 9:     end for
  - 10:  end for
  - 11: end for
- 

- Back-filling Process Let  $y_{\epsilon_{last}}$  be the erasures at the last flagged equations. The back-filling algorithm is done starting from the last row and solving the erasures backwards. Designate the last processed erasure as  $y_{\epsilon_{last}}$  and  $Num(\epsilon)_i$  as an erasure counter for the  $i$ -th parity-check polynomial equation. The function of  $Pos(y_{\epsilon_s})$  returns the position regarding to the parity-check polynomial of the  $x$ -th erasure. If and only if  $h_l(Pos(y_{\epsilon_{last}})) = 1$  and  $Num(\epsilon)_l = 1$ ,

$$y_{\epsilon_{last}} = \sum_{ct=0, Pos(y_{\epsilon_{last}})}^{n-1} h_l \oplus (ct)y(ct) \quad (2.13)$$

## 2.4 Matrix-based In-place Decoder

Then, the second to last row is processed in a similar manner. After a completion of the Back-filling process, the decoded sequence should be the original transmitted data  $x$ .

The schematic diagram of the In-place algorithm is illustrated in Figure 2.8, which clearly exhibits two process blocks: the upper block is to function as the Polynomial-update process and the lower block is to work as the back-filling process.

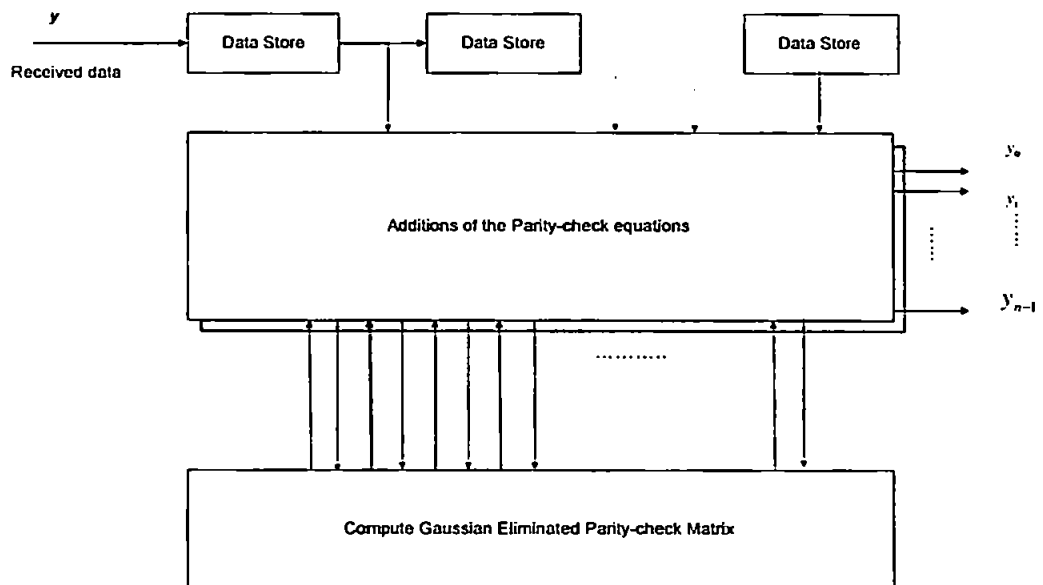


Figure 2.8: Matrix-based In-place Algorithm

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

**Example 2.3:** This example is to work on the BCH (15,7) code with a  $d_{\min}$  of 5. This code is guaranteed to correct 4 erasures by using syndrome decoding [37]. Its parity-check matrix is given as

$$H_{\text{bch}(15,7)} = \begin{pmatrix} 110100010000000 \\ 011010001000000 \\ 001101000100000 \\ 000110100010000 \\ 000011010001000 \\ 000001101000100 \\ 000000110100010 \\ 000000011010001 \end{pmatrix}$$

If the received sequence  $\mathbf{y}$  with erasures in position  $\{1, 2, 3, 4\}$ , referring to the  $\mathbf{H}$ , we can have two sets of equations:

- *the received set of equations* \*\*

$$y_{\text{rec}}(5) + y_{\text{rec}}(6) + y_{\text{rec}}(8) + y_{\text{rec}}(12) = 0$$

$$y_{\text{rec}}(6) + y_{\text{rec}}(7) + y_{\text{rec}}(9) + y_{\text{rec}}(13) = 0$$

$$y_{\text{rec}}(7) + y_{\text{rec}}(8) + y_{\text{rec}}(10) + y_{\text{rec}}(14) = 0$$

- *the erased set of equations*

$$y_{\text{rec}}(0) + y_{\epsilon}(1) + y_{\epsilon}(3) + y_{\text{rec}}(7) = 0 \quad (2.14)$$

$$y_{\epsilon}(1) + y_{\epsilon}(2) + y_{\epsilon}(4) + y_{\text{rec}}(8) = 0 \quad (2.15)$$

$$y_{\epsilon}(2) + y_{\epsilon}(3) + y_{\text{rec}}(5) + y_{\text{rec}}(9) = 0 \quad (2.16)$$

$$y_{\epsilon}(3) + y_{\epsilon}(4) + y_{\text{rec}}(6) + y_{\text{rec}}(10) = 0 \quad (2.17)$$

$$y_{\epsilon}(4) + y_{\text{rec}}(5) + y_{\text{rec}}(7) + y_{\text{rec}}(11) = 0 \quad (2.18)$$

---

\*\*For convenience, the expression of  $y(i)$  represents the received bit at  $i^{\text{th}}$  position.

## 2.4 Matrix-based In-place Decoder

---

Polynomial-update Process These two sets are in response to  $H_{rec}$  and  $H_c$ . Starting with  $y_c(1)$ , Equation (2.14) is flagged and subtracted from (2.15), only because  $y_c(1)$  is not contained in the other equations. Then the erased set can be written as:

$$y_{rec}(0) + y_c(1) + y_c(3) + y_{rec}(7) = 0 \text{ * for } y_c(1) \quad (2.19)$$

$$y_{rec}(0) + y_c(3) + y_{rec}(7) + y_c(2) + y_c(4) + y_{rec}(8) = 0 \quad (2.20)$$

$$y_c(2) + y_c(3) + y_{rec}(5) + y_{rec}(9) = 0 \quad (2.21)$$

$$y_c(3) + y_c(4) + y_{rec}(6) + y_{rec}(10) = 0 \quad (2.22)$$

$$y_c(4) + y_{rec}(5) + y_{rec}(7) + y_{rec}(11) = 0 \quad (2.23)$$

The \* represents the flagging of (2.19) meaning that this equation will be fixed and used to solve for  $y_c(1)$ .

The next unknown is  $y_c(2)$  contained in unflagged (2.20). This equation is flagged and subtracted from the non flagged equations containing  $y_c(2)$  to produce the next set of equations.

$$y_{rec}(0) + y_c(1) + y_c(3) + y_{rec}(7) = 0 \text{ * for } y_c(1) \quad (2.24)$$

$$y_{rec}(0) + y_c(3) + y_{rec}(7) + y_c(2) + y_c(4) + y_{rec}(8) = 0 \text{ * for } y_c(2) \quad (2.25)$$

$$y_{rec}(0) + y_{rec}(7) + y_c(4) + y_{rec}(8) + y_{rec}(5) + y_{rec}(9) = 0 \quad (2.26)$$

$$y_c(3) + y_c(4) + y_{rec}(6) + y_{rec}(10) = 0 \quad (2.27)$$

$$y_c(4) + y_{rec}(5) + y_{rec}(7) + y_{rec}(11) = 0 \quad (2.28)$$

The next unknown is  $y_c(4)$  contained in (2.26). This equation is flagged and subtracted from the non-flagged equations containing  $y_c(4)$  to produce the next set of equations.



## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

$$y_{rec}(0) + y_c(1) + y_c(3) + y_{rec}(7) = 0 \text{ * for } y_c(1) \quad (2.29)$$

$$y_{rec}(0) + y_c(3) + y_{rec}(7) + y_c(2) + y_c(4) + y_{rec}(8) = 0 \text{ * for } y_c(2) \quad (2.30)$$

$$y_{rec}(0) + y_{rec}(7) + y_c(4) + y_{rec}(8) + y_{rec}(5) + y_{rec}(9) = 0 \text{ * for } y_c(4) \quad (2.31)$$

$$y_c(3) + y_{rec}(7) + y_{rec}(8) + y_{rec}(5) + y_{rec}(9) + y_{rec}(6) + y_{rec}(10) = 0 \quad (2.32)$$

$$y_{rec}(0) + y_{rec}(8) + y_{rec}(9) + y_{rec}(11) = 0 \quad (2.33)$$

Back-filling Process There is now only one unknown remaining in an unflagged equation, which is  $y_c(3)$  in (2.32). This is solved first to find  $y(3)$  which is substituted into all flagged equations that  $y_c(3)$  appears, i.e. Equation (2.29) and (2.30). Equation (2.31) is solved next for  $y_c(4)$  to determine  $y(4)$  which is then substituted into the (2.30). Equation (2.30) is solved next for  $y_c(2)$  and finally Equation (2.29) is solved for  $y_c(1)$ .

---

It is worthy to compare the In-place algorithm with the well-established algebraic decoding algorithm. Given a binary  $(n, k, d = 2t + 1)$  BCH code, the algebraic decoder is capable of decoding up to  $2t = d - 1$  erasures, while the In-place algorithm is capable of decoding up to  $n - k (\geq d)$  erasures. Also, the distance distribution of BCH codes is binomial distributed. Therefore, the In-place algorithm can decode more erasures than the algebraic decoding algorithm for the most of time.

For a quick comparison on the performance of the recovery decoder, the guess/multi-guess decoder and the In-place decoder, we also use the PEG LDPC code (256, 128) as the code candidate. Figure 2.9 exhibits the In-place decoder achieves the best performance especially at a low  $p$  region. When  $p = 0.35$ , the performance of the In-place decoder is at least 100 times better than those of the Recovery algorithm and the S-Guess algorithm ( $ng_{max} = 3$ ), around 50 times better than that of the C-Guess algorithm ( $ng_{max} = 3$ ).

## 2.4 Matrix-based In-place Decoder

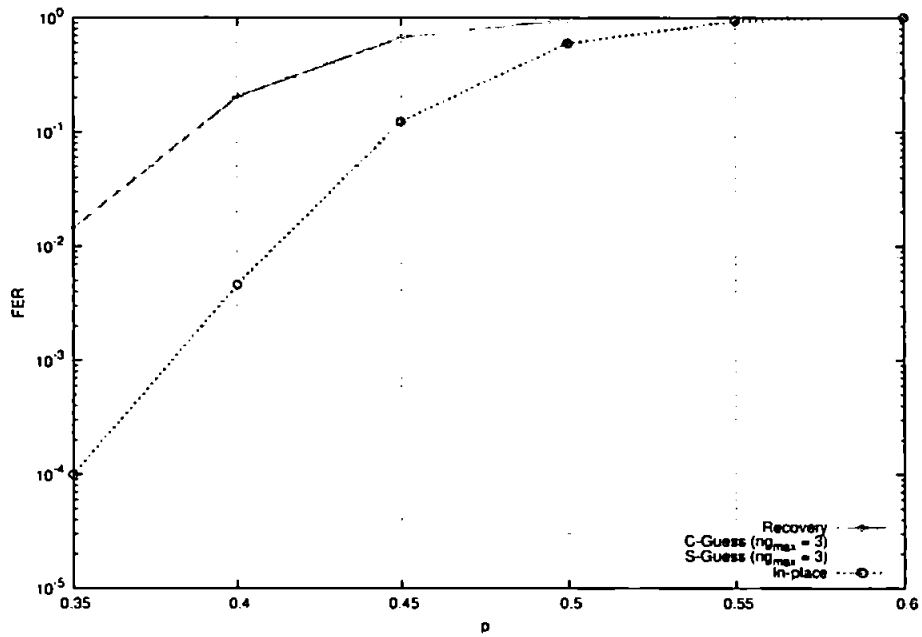


Figure 2.9: A performance comparison on the results of the Recovery decoder, Guess decoder (including C-Guess and S-Guess) and the In-place decoder

## 2.5 Numerical Results and Discussion

### 2.5.1 Performance of the Recovery Algorithm

First, we evaluate the performance of the Recovery Algorithm with the LT codes with Soliton distribution as described in [38] and irregular LDPC codes. As shown in Figure 2.10, the performance of irregular LDPC codes is significantly better than that of the LT codes for the same block length. As a consequence, we mainly use LDPC codes to benchmark the remaining algorithms.

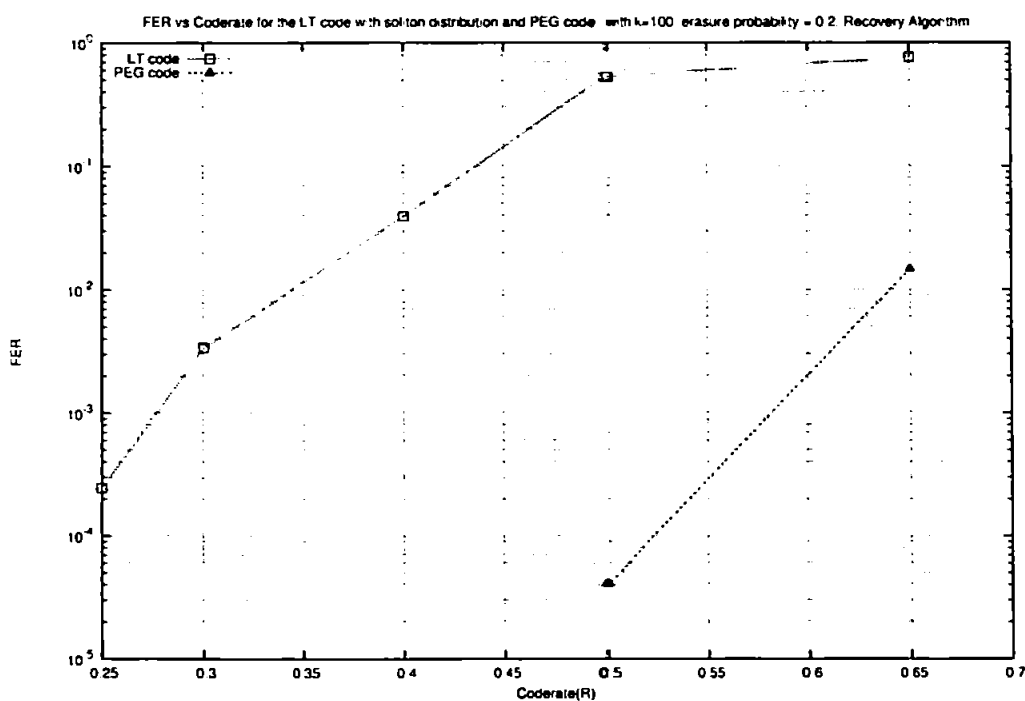


Figure 2.10: Performance of the LT codes and irregular LDPC codes with the erasure probability  $p = 0.2$

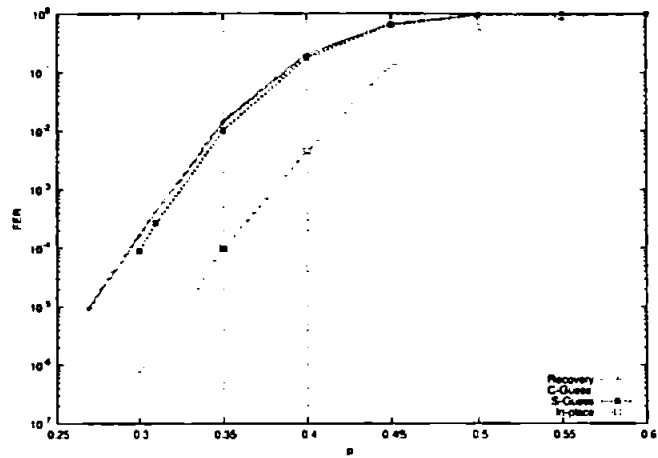
A particularly strong binary code having a sparse  $H$  is the cyclic LDPC code (255, 175), which has a length of 255 bits after encoding of 175 information bits. The cyclic LDPC code (255, 175) has a minimum Hamming distance of 17. Since

## 2.5 Numerical Results and Discussion

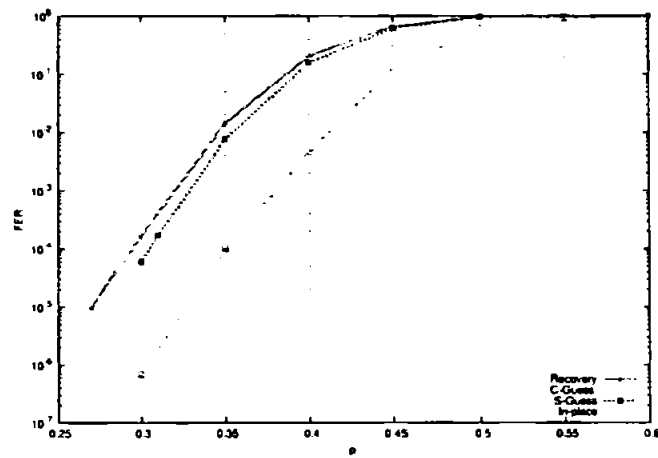
---

One-step majority-logic (OSML) codes have been rediscovered [35] because of their good performance as LDPC codes with iterative decoding algorithms. The parity-check equations of OSML codes are orthogonal on each bit position of the codeword which implies the absence of cycles of length 4 [73]. We apply different guess algorithms on the (255, 175) OSML code. From Figure 2.14, we can see that the crucial guess algorithm has the same performance as the sequential guess algorithm, whereas the crucial guess algorithm works effectively mainly for LDPC codes.

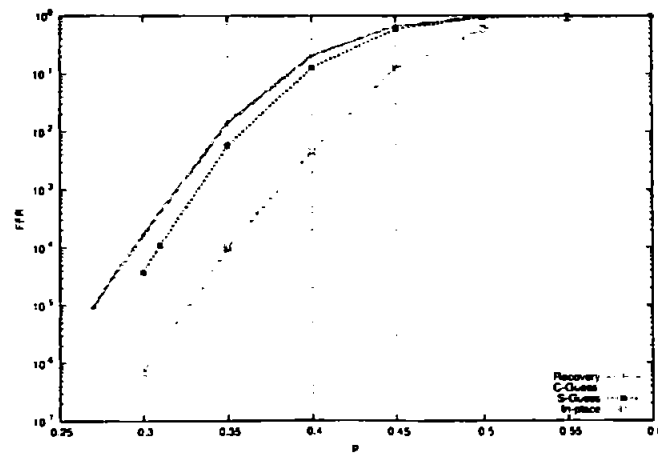
## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS



(a)  $ng_{\max} = 1$



(b)  $ng_{\max} = 2$



(c)  $ng_{\max} = 3$

Figure 2.12: Different guess algorithms for the PEG LDPC (256,128)

the parity-check polynomial of the (255, 175) code is orthogonal on every bit position, the minimum Hamming distance is  $1 + w$ , where  $w$  denotes the number of ones per row in  $H$  [53].

### 2.5.2 Performance of the Guess/Multi-Guess Algorithms

We then investigate the performance of the guess/multi-guess algorithms on LDPC codes and one-step majority-logic (OSML) codes.

Figure 2.11 shows the comparison of the performance between the recovery algorithm and the guess algorithm for the (255, 175) code. Also in this graph, we give the performance of the In-place algorithm which can be considered as the ML performance for the (255, 175) code. Due to the sparse structure of its parity-check matrix, the guess algorithm with less than 3 guesses can achieve more than 1 order of magnitude improvement compared to that of the recovery algorithm. It is also illustrated in Figure 2.11, the curve of the guess algorithm is very close to the curve of the In-place algorithm, which implies the guess algorithm is a “sub-optimal” decoding algorithm when the applied code is constructed by a sparse parity-check matrix.

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

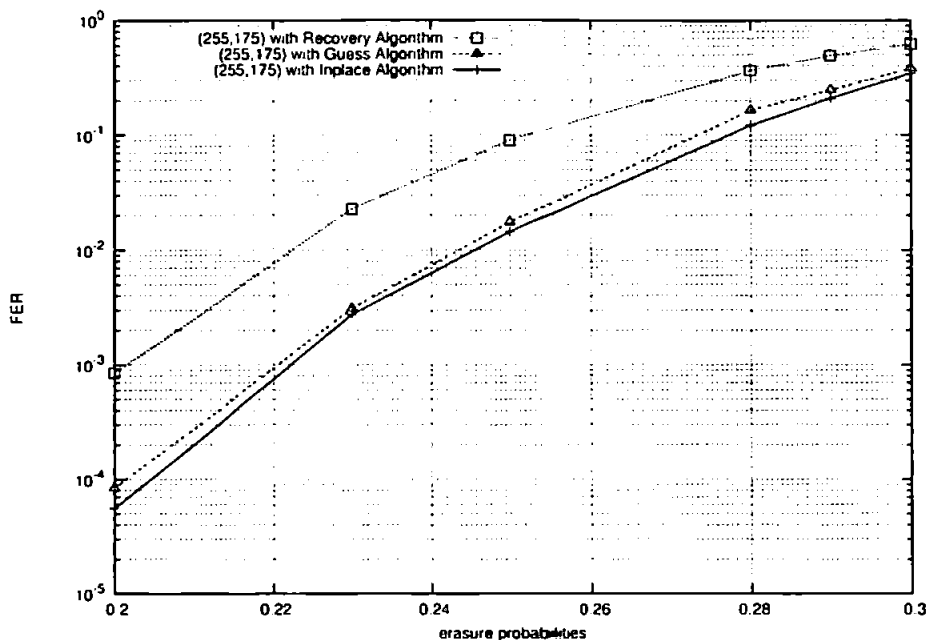
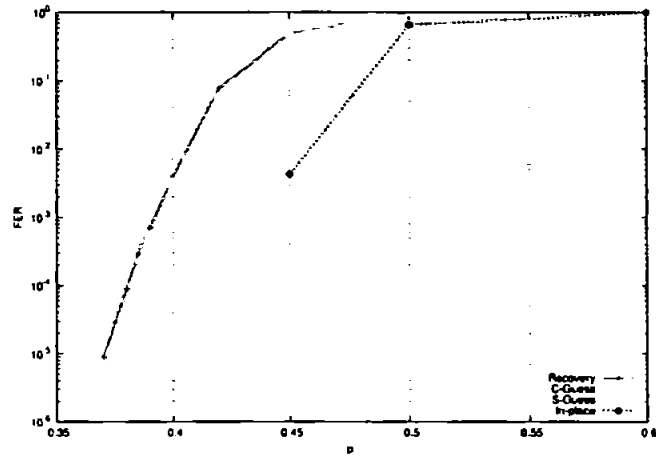


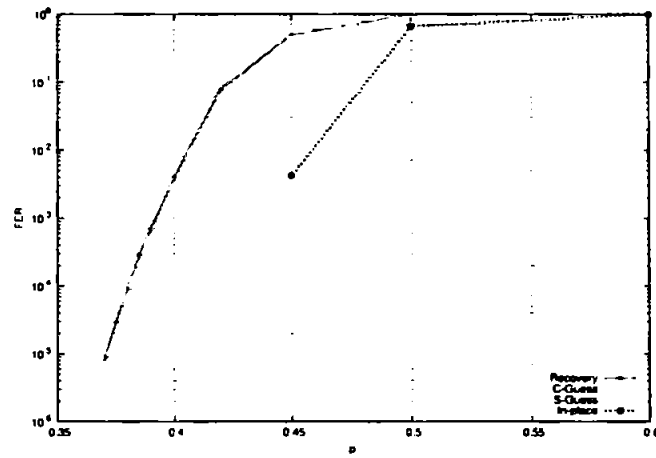
Figure 2.11: Performance of the Cyclic LDPC (255,175) with the Guess and In-place algorithm

Figures 2.12 and 2.13 depict the difference between the sequential guess algorithm and the crucial guess algorithm on LDPC codes. As can be seen, the more the number of guesses, the better the performance of the guess algorithm can be obtained. When the number of guesses is more than 1, the superiority of the crucial guess algorithm in comparison to the sequential guess algorithm is obviously presented in both figures. Especially in the graph of the PEG LDPC (256, 128), when  $ng_{\max} = 3$ , the crucial guess algorithm achieves the performance asymptotically close to the ML performance obtained by the In-place algorithm as the erasure probability decreases.

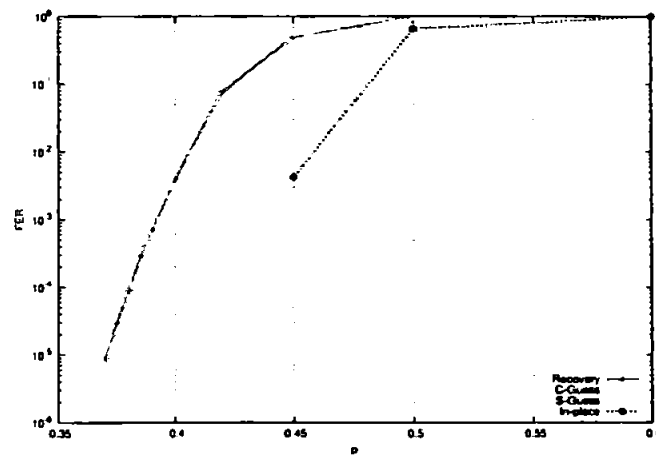
## 2.5 Numerical Results and Discussion



(a)  $ng_{\max} = 1$



(b)  $ng_{\max} = 2$

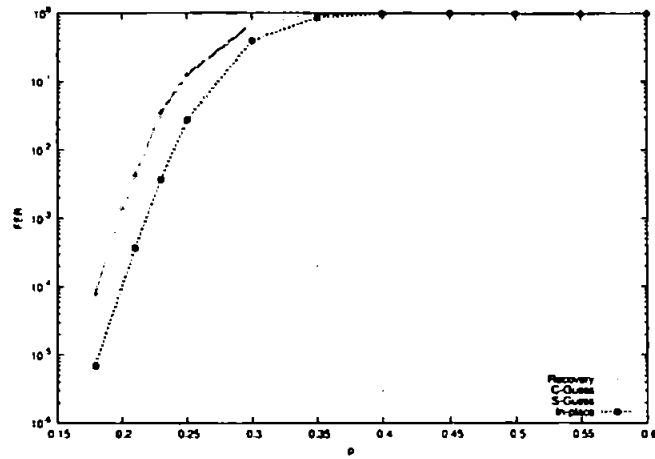


(c)  $ng_{\max} = 3$

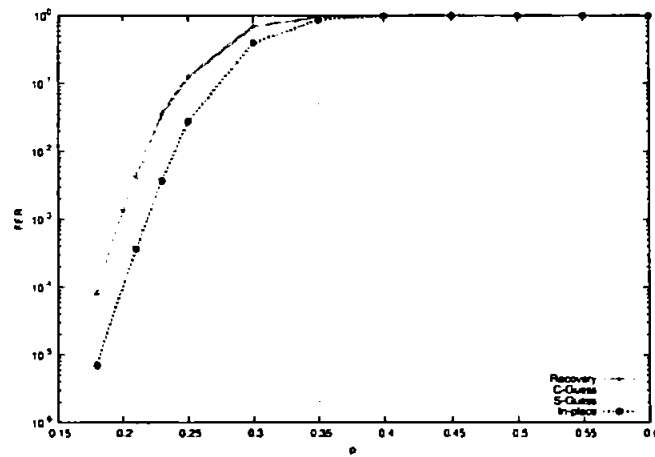
Figure 2.13: Different guess algorithms for the LDPC (1024,512,24)



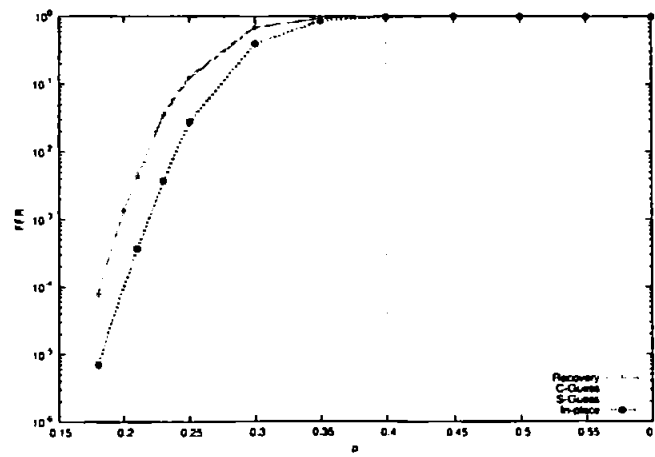
## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS



(a)  $ng_{\max} = 1$



(b)  $ng_{\max} = 2$



(c)  $ng_{\max} = 3$

Figure 2.14: Different guess algorithms for the OSML(255,175)

### 2.5.3 Performance of the In-place Algorithm

Figure 2.15 shows the performance of the (341, 205) LDPC code, which has a minimum Hamming distance of 16. Comparing these results of the recovery and the guess algorithms, the multi-guess algorithm can obtain the results by several orders of magnitude better. For example, when the erasure probability equals 0.3, the multi-guess algorithm with  $ng_{\max} = 3$  is one order of magnitude better than the recovery and guess algorithms; when  $ng_{\max} = 5$ , the multi-guess algorithm is 2 orders of magnitude better than the recovery and guess algorithms. Significantly, as an MLD, the In-place algorithm can achieve 4 orders of magnitude better than the recovery and guess algorithms.

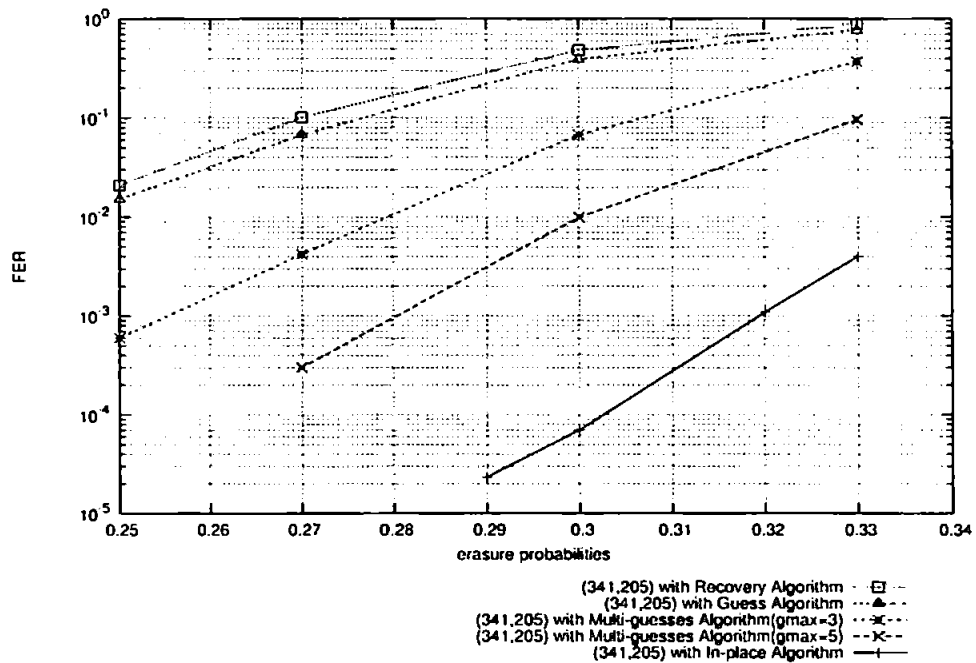


Figure 2.15: Performance of the Cyclic LDPC (341,205) with the Recovery, the Guess, the Multi-Guess and the In-place Algorithms

The ultimate performance of the In-place algorithm as a function of error correcting code is shown in Figure 2.16 for the example (255, 175) code which can correct a maximum of 80 erased bits. Figure 2.16 shows the probability density

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

function of the number of erased bits short of the maximum correctable which is  $n - k$ .

The results were obtained by computer simulations. The probability of being able to correct only 68 bits, a shortfall of 12 bits, is  $1.1 \times 10^{-3}$ . Simulations indicate that on average 77.6 erased bits may be corrected for this code. In comparison the BCH (255,178) code having similar rate is also shown in Figure 2.16. The BCH code has a similar rate but a higher minimum Hamming distance of 22 (compared to 17). It can be seen that it has better performance than the (255,175) code but it has a less sparse parity check matrix and consequently it is less suitable for the recovery algorithm and the guess algorithm. Moreover the average shortfall in erasures not corrected is virtually identical for the two codes.

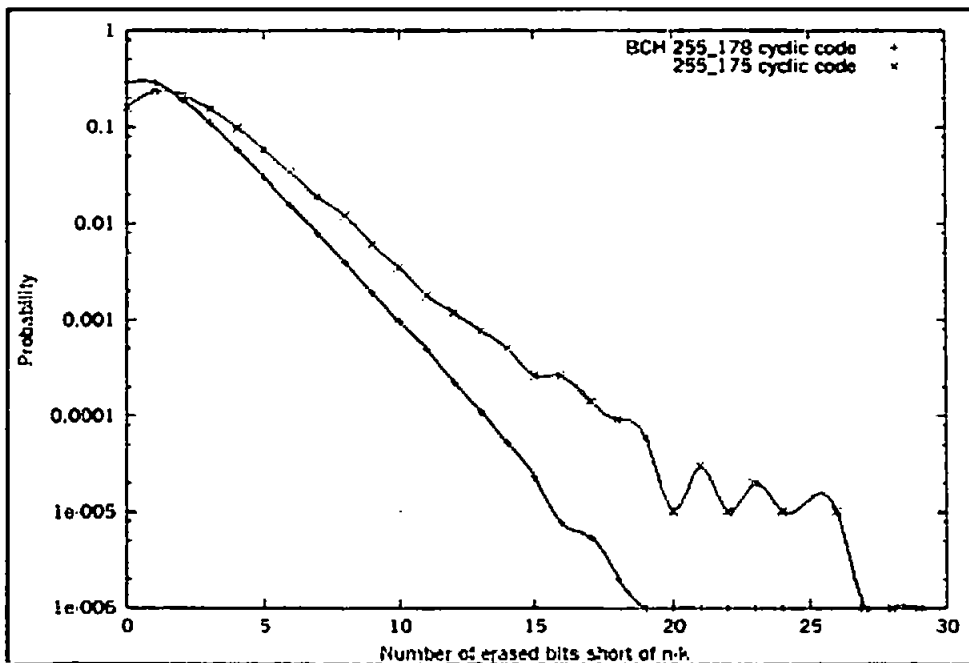


Figure 2.16: Comparison of Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $n - k$ ) for (255,175) code and BCH (255,178) code

The simulation results of using In-place Algorithm for the (103, 52) quadratic residue binary code [43] are shown in Figure 2.17. The minimum Hamming distance for this code is 19 and the results are similar to that of the (255, 178)

## 2.5 Numerical Results and Discussion

BCH code above. It is found from the simulations that on average 49.1 erasure bits are corrected (out of a maximum of 51) and the average shortfall from the maximum is 1.59 bits.

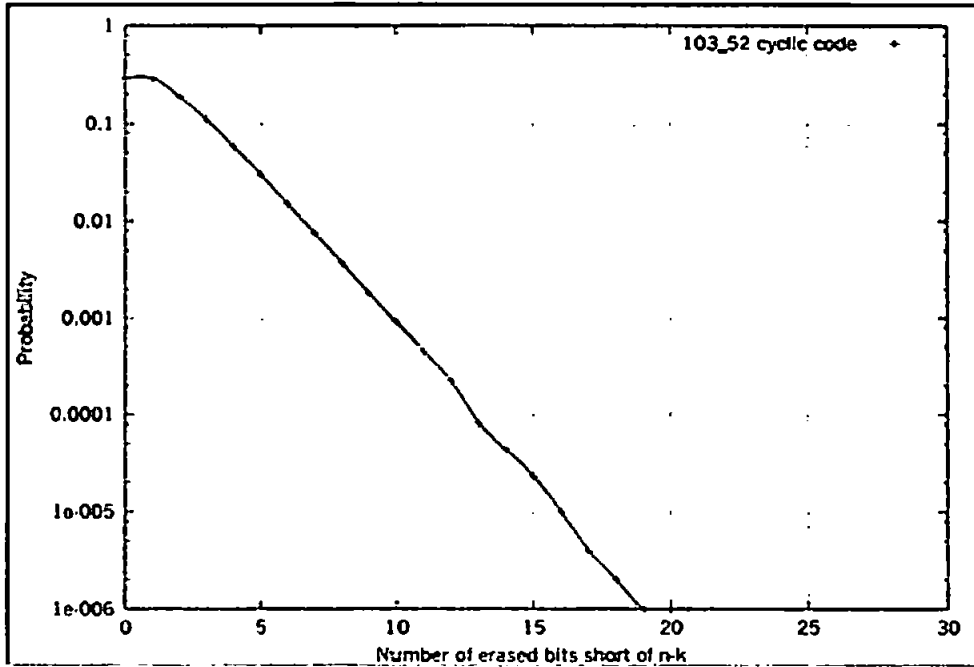


Figure 2.17: Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $n - k$ ) for (103, 52) code quadratic residue code

Similarly the results for the extended BCH (128,64) code is shown in Figure 2.18. This code has a minimum Hamming distance of 22 and has a similar probability density function to the other BCH codes above. On average 62.39 erasure bits are corrected (out of a maximum of 64) and the average shortfall is 1.61 bits from the maximum.

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

---

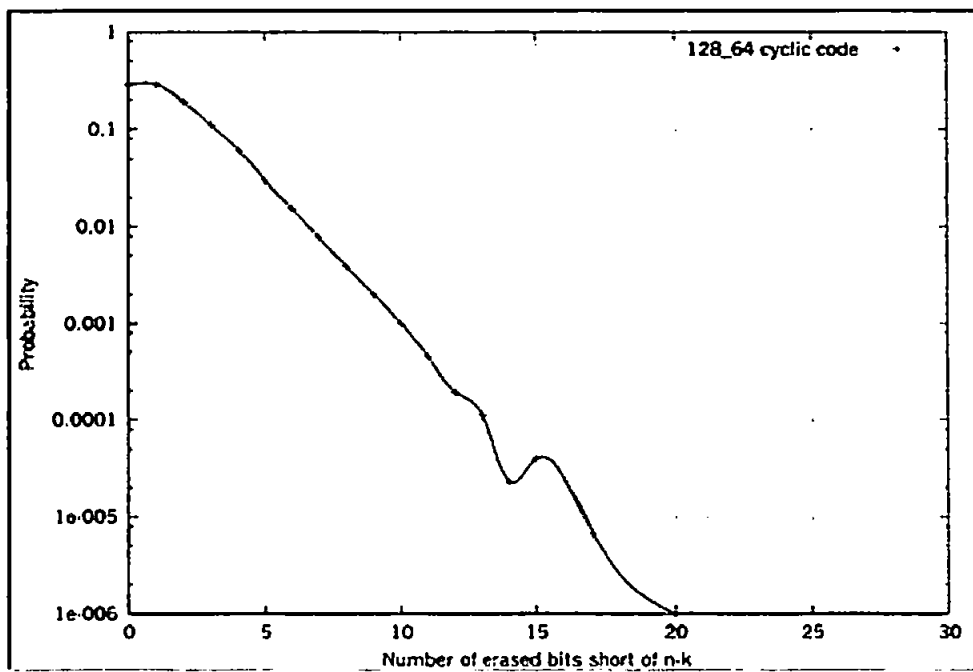


Figure 2.18: Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $n - k$ ) for (128,64) extended BCH code

## 2.6 Summary

In this chapter, we present different decoding algorithms of erasure codes, especially for LDPC codes over the BEC: the recovery, guess/multi-guess and the In-place algorithm. To break the stopping sets caused by the code structure effectively and efficiently, we propose the crucial guess algorithm. The multi-guess algorithm is an extension to the guess algorithm, which can push the limit to break the stopping sets. We also show that the guess and multi-guess algorithms are parity-check matrix dependent. For the codes with sparse parity-check matrix, the guess/multi-guess algorithms can be considered as “sub-optimal decoding” algorithms, in particular with the crucial guesses.

The In-place algorithm is an optimal method which is capable of achieving an ML performance for the BEC. For linear block codes, the In-place algorithm ensures to correct  $n - k - \rho$  erasures, where  $\rho$  is a small positive integer.

Chapter 5 and Chapter 6 will further optimise the application of the In-place algorithm with a much reduced computational complexity.

## 2. MATRIX-BASED ERASURE DECODING ALGORITHMS

*Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth.*

Arthur Conan Doyle (1859-1930)

# 3

## Branch-Evaluation Search on the Code-Tree Algorithm

### 3.1 Background

In this chapter, we consider the performance of (erasure) codes on the Additive White Gaussian Noise (AWGN) channel. When erasure codes are employed for different communication systems, the outputs for decoders are no more three levels (0, 1, "?"), which are so-called "soft-decision" outputs. By deploying the soft-decision outputs, the code performance is about 3 dB of coding gain [37] over the hard-decision algorithm. The better performance is paid by more computational complexity. The decoding algorithms with the use of the soft-decision received sequence are generally called *soft-decision decoding*, which normally are categorised as *reliability-based* decoding algorithms and *code structure-based* decoding algorithms. The *ordered statistic decoding* (OSD) algorithm [23] has been known as an ML-approachable algorithm which is implemented based on



### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

the channel reliabilities of received vectors. It is capable of offering an ML performance when the number of orders is up to the information length of the applied code. Under this situation, the OSD algorithm is equivalent to its ancestor, the classic *Dorsch* decoding algorithm [17], in which, hard decisions are derived from the soft decisions using standard bit-by-bit detection, choosing the binary state closet to the received coordinate. The idea of the *Dorsch* algorithm has been extended by Tomlinson *et al.* [75], which involves a technique for any linear  $(n, k)$  code, erasures that  $(n - k)$  less reliable, soft decisions of each received vector may be treated as erasures in determining candidate codewords.

As first introduced in computer science, the tree structure, emulating a traditional hierarchical structure by a set of branches and vertices, has been widely introduced to the hierarchical data manipulation, the information search engine, the special composition of digital images, the industrial system safety evaluation and so on.

By utilising the hierarchical property of the tree structure, in 2005 and 2006, Rosnes & O.Ythelhus [62], Rosnes & Ytrehus [63], proposed a precise but exhaustive tree-structured search algorithm on turbo codes' stopping sets, which has been known as "Exhaustive Tree-Search Algorithm". In 2007, Rosnes & Ytrehus [64] upgraded the algorithm and employed it to find all the stopping sets and codewords for LDPC codes. The upgraded tree-search algorithm improves the search efficiency by giving a preset limitation on the search scope. In 2009, Ambroze *et al.* [2] further developed the tree search algorithm into a tree-based weight spectrum evaluation algorithm.

In this chapter, we propose a new universal decoding algorithm which is a combination of the reliability-based decoding algorithm and the code structure-based decoding algorithm. This chapter starts with a review of the classic reliability-based algorithm – the OSD decoding algorithm to discuss the crucial metrics to approach an ML performance in Section 3.2. Section 3.3 gives a description on the code tree and defines different types of code trees based on their growing directions. In Section 3.4, we describe the algorithm in details. Then, we apply our decoding algorithm to different linear codes and compare the performance and computational complexity with the OSD algorithm in Section 3.5.

## 3.2 A Review of the Ordered Statistical Decoding Algorithm

The OSD algorithm [23], proposed in 1995, has been considered as an ML-approachable reliability-based decoding algorithm.

Assume a BPSK-modulated vector  $\mathbf{x} = \{x_0, x_1, \dots, x_{n-1}\}$  which has been encoded by the code  $\mathcal{C}(n, k)$ , is transmitted through the AWGN channel and then it is received as the received vector, defined as  $\mathbf{y} = \{y_0, y_1, \dots, y_{n-1}\}$ . The vector  $\mathbf{y}$  has its channel reliability, written as  $\mathcal{R}$ , as expressed in (3.1).

$$\begin{aligned} \mathcal{R} &= \{r_0, r_1, \dots, r_{n-1}\} \\ &= \{|y_0|, |y_1|, \dots, |y_{n-1}|\} \end{aligned} \tag{3.1}$$

Order the vector  $\mathbf{y}$  based on the reliability values in decreasing order into a new vector of  $\mathbf{y}'$ . And, the reliability vector  $\mathcal{R}$  is also permuted into a new reliability vector  $\mathcal{R}'$  with the same ordering rule as the one applied on  $\mathbf{y}$ .

$$\begin{aligned} \mathcal{R}' &= \{r'_0, r'_1, \dots, r'_{n-1}\} \\ &= \{|y'_0|, |y'_1|, \dots, |y'_{n-1}|\} \end{aligned} \tag{3.2}$$

where  $r'_0 > r'_1 > \dots > r'_{n-1}$ .

Denote the permutation function as  $\pi(\cdot)$ , and then we have:

$$\mathbf{y}' = \pi(\mathbf{y}) \tag{3.3}$$

$$\mathcal{R}' = \pi(\mathcal{R}) \tag{3.4}$$

Applying the  $\pi(\cdot)$  function on the columns of the generator matrix  $\mathbf{G}$ , we will have

$$\mathbf{G}' = \pi(\mathbf{G}) \tag{3.5}$$

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

If the first  $k$  columns are not independent \* and so they are not able to represent the information part of the codeword, the second permutation function  $\pi_2(\cdot)$  is required to put  $\mathbf{G}'$  as a systematic generator matrix  $\tilde{\mathbf{G}}$  and therefore the received sequence is further re-ordered as

$$\begin{aligned}\tilde{\mathbf{y}} &= \pi_2(\mathbf{y}') \\ &= \pi_2(\pi(\mathbf{y}))\end{aligned}\quad (3.6)$$

Decode the sequence  $\tilde{\mathbf{y}}$  from (3.6) by a *hard-decision decoder* (HDD) and then output a codeword, denoted as  $\tilde{\mathbf{z}}$ .

$$\tilde{\mathbf{z}} = \underbrace{(\tilde{z}_0, \tilde{z}_1, \dots, \tilde{z}_{k-1})}_{\text{MRB}}, \underbrace{(\tilde{z}_k, \dots, \tilde{z}_{n-1})}_{\text{LRB}}\quad (3.7)$$

The first  $k$  components of  $\tilde{\mathbf{z}}$  are the so-called *most reliable basis* (MRB) part and the other  $n - k$  components are defined as the *least reliable basis* (LRB) part for the applied code  $\mathcal{C}$ . Because the MRB part contains the bits/symbols with the most reliability, it should have few errors.

Denote  $\tilde{\mathbf{z}}_k$  as the set of the MRB bits. Encode  $\tilde{\mathbf{z}}_k$  with the systematic generator matrix  $\tilde{\mathbf{G}}$ , a corresponding codeword  $\tilde{\mathbf{c}}$  is obtained.

$$\tilde{\mathbf{c}} = \tilde{\mathbf{z}}_k \tilde{\mathbf{G}}\quad (3.8)$$

By de-permuting the codeword with the permutation functions  $\pi(\cdot)$  and  $\pi_2(\cdot)$ , a potential codeword  $\hat{\mathbf{c}}$  from the original code  $\mathcal{C}$  is obtained by:

$$\hat{\mathbf{c}} = \pi^{-1}(\pi_2^{-1}(\tilde{\mathbf{c}})).\quad (3.9)$$

By then, the process of the OSD-0 decoding is completed. When the order of the OSD decoding is more than zero, the additional process is performed. Suppose that an OSD decoding is with its order of  $l$ . For the variable  $l$  in  $0 \leq l \leq t$ , make all possible changes of  $l$  of  $\tilde{\mathbf{z}}_k$  and then encode these changed sequences into

---

\* After applying the Gaussian Elimination algorithm on the matrix, if the rank of the identity part equals to  $k$ , we say the first  $k$  columns are independent; otherwise, the first  $k$  columns are not linearly independent.

### 3.3 Code Tree Representation of Linear Block Codes

---

codewords  $\{\hat{c}_1, \hat{c}_2, \dots, \hat{c}_{Num(OSD_t)-1}\} = \hat{\mathcal{C}}$ . The notation of  $\hat{\mathcal{C}}$  represents a potential codeword set.

The number of the potential codewords in  $\hat{\mathcal{C}}$  can be derived by:

$$Num(OSD_t) = \sum_{j=0}^t \binom{k}{j} \quad (3.10)$$

Then compute the cross-correlation metric  $correl(\tilde{y}, \hat{c}_j)$  as follows:

$$correl(\tilde{y}, \hat{c}_j) = \sum_{s=0}^{n-1} \tilde{y}_s \hat{c}_{j,s} \quad (3.11)$$

The potential codeword with the maximum correlation result is selected as the decoded codeword.

Obviously, the OSD algorithm is an ML-approachable algorithm when  $l = k$ . It is shown in [23] that for most block codes of lengths up to 128 and rates  $k/n \geq \frac{1}{2}$  that  $t = \lfloor d_{\min}/4 \rfloor$  is capable of achieving an ML decoding performance.

### 3.3 Code Tree Representation of Linear Block Codes

The code tree is a graphical code representation by tree structure, which is composed of branches and vertices. The set of branches is written as  $\mathbf{B} = \{B_0, B_1, \dots\}$  and the set of vertices is written as  $\mathbf{V} = \{v_0, v_1, \dots, v_n\}$ . The arborescence traits can be represented mathematically by a given specific order, which is controlled by the code. Consider a binary code  $\mathcal{C}(n, k)$  with its generator matrix and parity-check matrix,  $\mathbf{G}$  and  $\mathbf{H}$  respectively. Each codeword  $c_i \in \mathcal{C}$  constitutes a complete branch  $B_i$  of its corresponding code tree  $\mathcal{T}_{\mathcal{C}}$ , where the index  $i$  is in the range of  $0 \leq i \leq 2^k - 1$  †. Therefore, a complete code tree should contain  $2^k$  branches.

As an instance, the complete branch of the all-zero codeword is  $B_0$  consists  $n$  spans, which are connected by  $n + 1$  vertices, which is shown in Figure 3.1. Each span is denoted as  $b_{i,j}$ . The value of  $b_{i,j}$  also represents the label on this span.

---

†Normally,  $i$  is the decimal representation of the codeword sequence.

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

For a codeword  $c_i = (z_0 z_1 z_2 \dots z_{n-1})$  and its corresponding branch  $B_i = (b_{i,0} b_{i,1} b_{i,2} \dots, b_{i,n-1})$ , we have

$$\begin{aligned} b_{i,0} &= z_0 \\ b_{i,1} &= z_1 \\ &\vdots \\ b_{i,n-1} &= z_{n-1} \end{aligned}$$

The set of vertices  $\{v_0, v_1, v_2, \dots, v_n\}$  can also be considered as the indication of the tree growing direction. At each vertex, the branch diverges into  $q$  spans when the applied code is over  $GF(q)$ . In this chapter, we only consider binary linear codes and henceforth, at each vertex, there are at most 2 diverged spans. The vertices of  $v_0$  and  $v_n$  represent the ends of the tree. Different to the trellis representation of a code, the code tree does not have to start from the vertex of  $v_0$ . It can be initialised from any intermediate vertex  $v_j$ , where  $j$  can be arbitrarily chosen from 0 to  $n$ . The initial growing vertex is designated as  $v_{initial}$ . The first example of code tree as shown in Figure 3.1 is with its  $v_{initial} = v_0$ . This kind of code tree is defined as *one-directional code tree*.

**3.1 Definition (One-directional Code Tree).** For a code tree representation, if the initial growing vertex  $v_{initial} = v_0$  or  $v_{initial} = v_n$ , the code tree expands only towards  $v_n$  or  $v_0$  respectively. This kind of code tree is called “one-directional” code tree.

The second example of code tree, given in Figure 3.2, shows a code tree growing from its  $v_{initial}$  at one of the intermediate vertex. This is one of the different points between the tree representation and the trellis representation. Instead of being controlled by the time only, the code tree is also under the control of the growing direction.

**3.2 Definition (Bi-directional Code Tree).** For a code tree representation, if the initial growing vertex  $v_{initial} = v_x$ , where  $0 < x < n$ , the code tree expands towards  $v_0$  or  $v_n$  simultaneously. This kind of code tree is defined as “bi-directional” code tree.

### 3.3 Code Tree Representation of Linear Block Codes

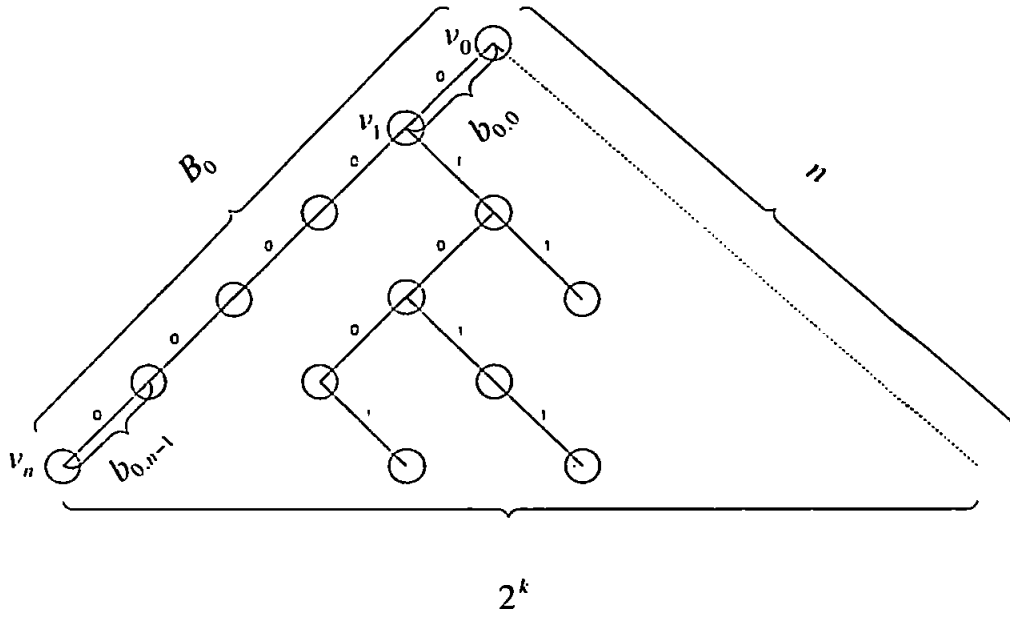


Figure 3.1: An example of Code Tree I (One-directional Code Tree)

As illustrated in Figure 3.2, the bi-directional code tree should have its both ends with the number of points more than 1 but less than or equal to  $2^k$ . This is controlled by the position of  $v_{initial}$ .

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

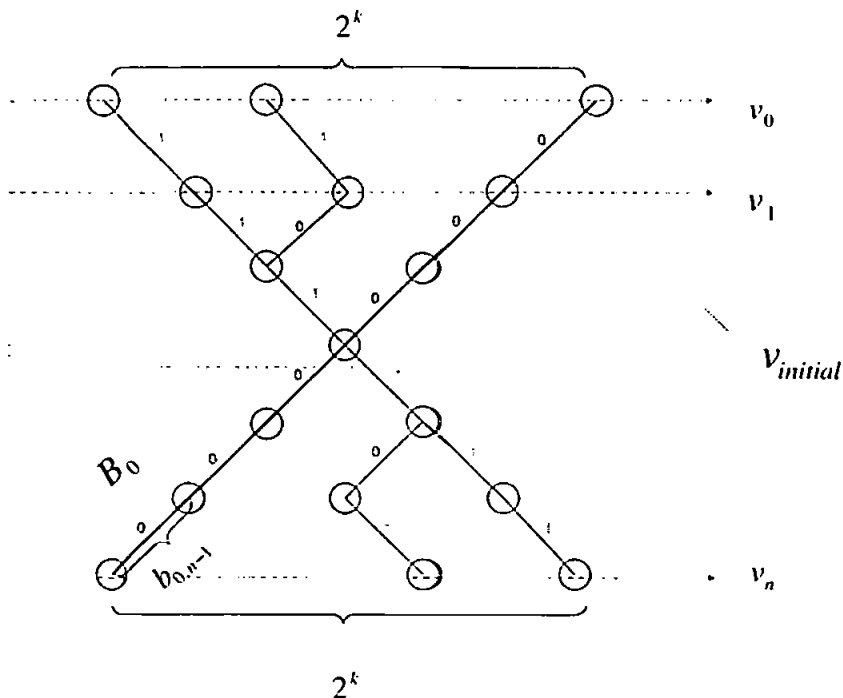


Figure 3.2: An Example of Code Tree II (Bi-directional Code Tree)

### 3.4 BESCT Algorithm

As proposed, the BESCT algorithm utilises the hierarchical property of the tree structure combined with the reliability-based decoding algorithm to realise an MLD. For the BESCT algorithm, we select the bi-directional code tree with its initial vertex at the separation of the information part and the parity-check part. To be more specific, we also call the initial vertex as the “cut point vertex”, written as  $v_{ct}$  and  $v_{ct} = v_{initial}$ . The specific code tree for the BESCT algorithm is designated as  $\mathcal{T}_{BESCT}$ .

Suppose the upper end in accordance with the parity-check part and the lower end in accordance with the information part. The number of vertices at the  $0^{th}$  upper end layer is  $2^{n-k}$  and the number of the vertices at the  $n^{th}$  lower end layer is  $2^k$ .

The cut point vertex  $v_{n-k}$  divides its  $\mathcal{T}_{BESCT}$  as the subtrees of  $\mathcal{T}(\mathbf{b}_0, \mathbf{b}_{n-k-1})$  and  $\mathcal{T}(\mathbf{b}_{n-k}, \mathbf{b}_{n-1})$ . The bold notation of  $\mathbf{b}_j$  presents all the branch spans connecting the vertices between the layer  $j$  and the layer  $j + 1$ . Obviously, the code tree  $\mathcal{T}_{BESCT}$  can be written as

$$\mathcal{T}_{BESCT} = \mathcal{T}(\mathbf{b}_0, \mathbf{b}_{n-k-1}) \circ \mathcal{T}(\mathbf{b}_{n-k}, \mathbf{b}_{n-1}) \quad (3.12)$$

In (3.12), the notation  $\circ$  represents the concatenation of the subtrees  $\mathcal{T}(\mathbf{b}_0, \mathbf{b}_{n-k-1})$  and  $\mathcal{T}(\mathbf{b}_{n-k}, \mathbf{b}_{n-1})$ .

Observing the  $i^{th}$  branch  $B_i$  corresponding to the codeword  $c_i$  it can also be written as

$$B_i = B'_i(b_{i,0}, b_{i,n-k-1}) \circ B'_i(b_{i,n-k}, b_{i,n-1}) \quad (3.13)$$

where we call  $B'_i$  as the sub-branch of the  $i^{th}$  branch. By setting  $v_{ct} = v_{n-k}$ , a complete branch is partitioned into a parity-check sub-branch and an information sub-branch, written as  $B'_i(b_{i,0}, b_{i,n-k-1})$  and  $B'_i(b_{i,n-k}, b_{i,n-1})$  respectively. Nevertheless, not every complete branch corresponds to a valid codeword. Therefore, to avoid an ambiguity to complete a valid codeword branch, at the  $v_{ct}$ , a coding rule, such as  $H$  or  $G$ , needs deploying.

---

**Example 3.1:** The Hamming code  $(7, 4, 3)$  has its systematic generator matrix  $G$  as follows:

$$G = \begin{pmatrix} 1101000 \\ 1010100 \\ 0110010 \\ 1110001 \end{pmatrix}. \quad (3.14)$$



### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

and then the corresponding full code book is listed as follows:

$$c = \left\{ \begin{array}{l} (000 \ 0000) \\ (110 \ 1000) \\ (111 \ 0100) \\ (011 \ 1100) \\ (011 \ 0010) \\ (101 \ 1010) \\ (000 \ 1110) \\ (111 \ 0001) \\ (001 \ 1001) \\ (010 \ 0101) \\ (100 \ 1101) \\ (100 \ 0011) \\ (010 \ 1011) \\ (001 \ 0111) \\ (111 \ 1111) \end{array} \right\}$$

Each row contains a single codeword with its LHS as the parity-check part and its RHS as the information part. Followed the content of the code book, the Hamming code (7, 4, 3) has its code tree  $\mathcal{T}_{BESCT}$  constructed as demonstrated in Figure 3.3.

In Figure 3.3, the blue-traced and red-traced spans have their labelled values of "0" and "1" respectively. Suppose the index of vertex ascending from the top to the bottom as  $\{v_0, v_1, \dots, v_7\}$  and the index of span ascending as  $\{b_{i,0}, b_{i,1}, \dots, b_{i,6}\}$ . Therefore,

$$\mathcal{T}_{BESCT} = \mathcal{T}(b_0, b_2) \circ \mathcal{T}(b_3, b_6)$$

$\mathcal{T}(b_0, b_2)$  is the parity-check subtree and  $\mathcal{T}(b_3, b_6)$  is the information subtree.  $\mathcal{T}(b_0, b_2)$  contains 8 sub-branches and  $\mathcal{T}(b_3, b_6)$  has 16 sub-branches.

---

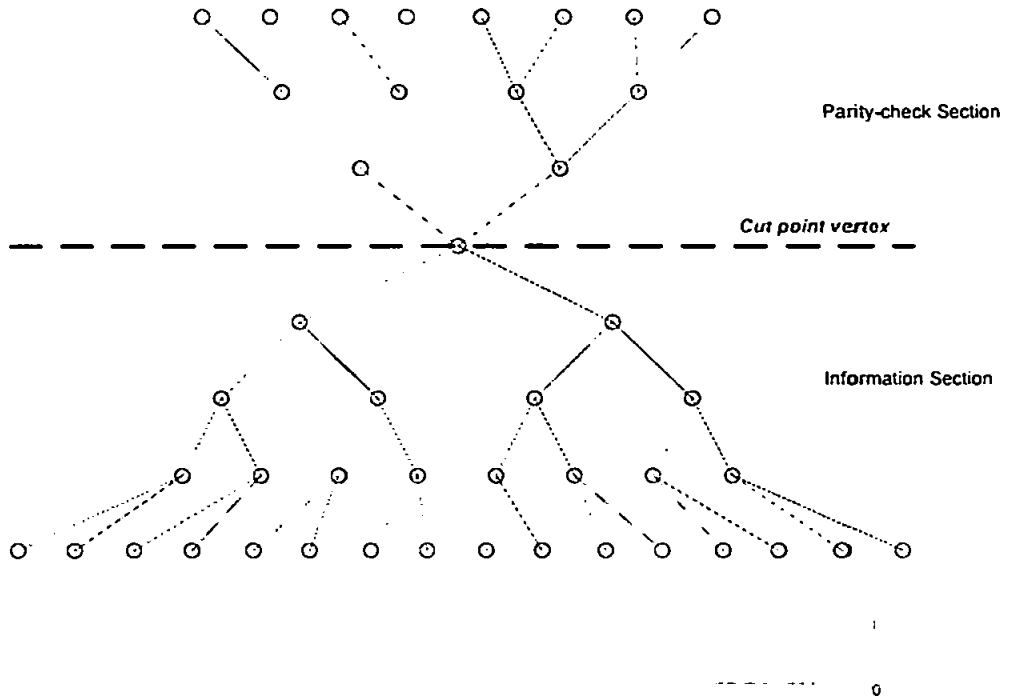


Figure 3.3:  $\mathcal{T}_{BESCT}$  of the Hamming code(7, 4, 3). A coding rule has to be applied to validate a codeword.

### 3.4.1 The Concept of Ensemble Branch

In this section, we introduce the concept of ensemble branch for an efficient BESCT process. For a  $\mathcal{T}_{BESCT}$  generated from  $C(n, k)$ , the code tree contains  $k$  ensemble branches  $\{B_0^{en}, B_1^{en}, \dots, B_{k-1}^{en}\}$ , each of which has the labels same as the elements of the generator polynomial  $g_i$  from the systematic generator matrix  $\mathbf{G}$ .

Then, we have:

$$\begin{aligned}
 B_i^{en} &= (b_{i,0}^{en} \ b_{i,1}^{en} \ \dots \ b_{i,n-1}^{en}) \\
 &= (g_{i,0} \ g_{i,1} \ \dots \ g_{i,n-1}) \\
 &= \mathbf{g}_i
 \end{aligned} \tag{3.15}$$

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

where  $i \in \{0, 1, \dots, k-1\}$ .

Using an instance in Example 3.1, for the Hamming code  $(7, 4, 3)$ , the  $\mathcal{T}_{BESCT}$  has the ensemble branches as

$$\begin{aligned} B_0^{en} &= (1101000) \\ B_1^{en} &= (1010100) \\ B_2^{en} &= (0110010) \\ B_3^{en} &= (1110001) \end{aligned} \quad (3.16)$$

By arbitrarily selecting and combining the ensemble branches, a valid branch can be generated. Whereas, the total number of the valid branches can be derived by:

$$Num(B) = \sum_{t=0}^k \binom{k}{t} \quad (3.17)$$

Therefore, for  $\mathcal{T}_{BESCT}$  of the Hamming code  $(7, 4, 3)$ , it has valid branches with the total number of  $\binom{4}{0} + \binom{4}{1} + \binom{4}{2} + \binom{4}{3} + \binom{4}{4} = 16$ .

#### 3.4.2 Cost and Dynamic Threshold

The  $\mathcal{T}_{BESCT}$  is constructed based on the ordered received vector  $\tilde{\mathbf{y}}$  as described in Section 3.2. Whereas, the parity-check subtree  $\mathcal{T}_{BESCT}(\mathbf{b}_0, \mathbf{b}_{n-k-1})$  corresponds to the LRB part and the information subtree  $\mathcal{T}_{BESCT}(\mathbf{b}_{n-k}, \mathbf{b}_{n-1})$  corresponds to the MRB part.

Instead of an exhaustive search over the complete  $\mathcal{T}_{BESCT}$ , the BESCT algorithm evaluates the possible branches by comparing their *costs* with a *dynamic threshold*.

**3.3 Definition.** The *cost*, written as  $cst(\cdot)$ , is designated as a cross-correlation metric between the branch/span labels and the ordered sequence  $\tilde{\mathbf{y}}$ .

- the cost of a single span:

$$cst(b_{i,q}) = (2b_{i,q} - 1)\tilde{y}_q \quad (3.18)$$

where  $0 \leq q \leq n - 1$ .

- the cost of a partial branch:

$$\begin{aligned} cst(B_i(b_{i,q_1}, b_{i,q_2})) &= (2b_{i,q_1} - 1)\tilde{y}_{q_1} + (2b_{i,(q_1+1)} - 1)\tilde{y}_{(q_1+1)} + \dots \\ &\quad + (2b_{i,(q_2-1)} - 1)\tilde{y}_{q_2-1} + (2b_{i,q_2} - 1)\tilde{y}_{q_2}; \end{aligned} \quad (3.19)$$

where  $0 \leq q_1 \leq q_2 \leq n - 1$ .

- the cost of a complete branch:

$$cst(B_i) = \sum_{j=0}^{n-1} (2b_{i,j} - 1)\tilde{y}_j, \quad (3.20)$$

where  $i$  is from 0 to  $2^k - 1$ .

The target of the BESCT algorithm is to search a branch with its cost closest to the overall magnitude of the received vector. Designate  $\text{Mag}(\cdot)$  as a function returning the sum of the magnitude values of a vector as follows:

$$\text{Mag}(\mathbf{y}) = \sum_{j=0}^{n-1} |y_j| \quad (3.21)$$

The  $\text{Mag}(\mathbf{y})$  is equivalent to a cross-correlation metric on the received vector  $\mathbf{y}$  and its hard-decision output  $\mathbf{u}$ . However, the hard-decision output  $\mathbf{u}$  may not be a valid codeword of  $\mathcal{C}$  if  $\mathbf{H} \cdot \mathbf{u}^T \neq \mathbf{0}$ . Then the BESCT algorithm starts to evaluate the branches from the spans connected to  $v_{ct}$ .

In the BESCT process, two dynamic thresholds control the evaluation process.

- The first threshold is based on current highest branch cost at stage  $i$ , which is designated as  $th_{cst}$  and called as the "cost threshold". Suppose at stage  $i$ , the cost threshold is  $th_{cst}$  and the cost of current branch is  $cst(B_i)$ .
- The second threshold is introduced as the term  $\delta(th_{cst}, cst(B_i))$ , which is the discrepancy between  $\text{Mag}(\mathbf{y})$  and  $cst(B_i)$ .  $\delta(\text{Mag}(\mathbf{y}), cst(B_i))$  is used

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

to control the *growing direction* of next branch and called as the “*direction threshold*”.

The concept of *growing direction* is devised to control the branch growing process, which contributes to a reduction of unnecessary computations and operations. From the cutting vertex  $v_{ct}$ , mark the vertices in the information tree as  $v_k^{in}, v_{k-1}^{in}, \dots, v_0^{in}$ . At the vertex of  $v_j^{in}$ , that is, at the  $j^{th}$  layer:

- ✓ the process of an inserting-vertex at the layer of  $j - 1$  is defined as *growing-forwards*, written as  $\mathcal{V}^+$ ;
- ✓ the process of a deleting-vertex at the layer of  $j$  is defined as *growing-backwards*, written as  $\mathcal{V}^-$ .

By utilising ensemble branches, the process of *branch growing* can be done in one step. For a given branch  $B_i = (b_{i,0}b_{i,1} \dots b_{i,n-1})$ , if an inserting-vertex or deleting-vertex process is to be executed at the  $j^{th}$  layer in the information sub-tree, the spans of the new branch  $B_{i+1}$  can be obtained as:

$$\begin{aligned}
 b_{i+1,0} &= b_{i,0} \oplus b_{j,0}^{en} \\
 b_{i+1,1} &= b_{i,1} \oplus b_{j,1}^{en} \\
 b_{i+1,2} &= b_{i,2} \oplus b_{j,2}^{en} \\
 &\vdots \\
 b_{i+1,n-1} &= b_{i,n-1} \oplus b_{j,n-1}^{en}
 \end{aligned}$$

Branch-wise, this operation can also be written as  $B_{i+1} = B_i \oplus B_j^{en}$ .

#### 3.4.3 The BESCT Algorithm

*Step 1* The information sub-branch  $B'_0(b_{0,n-k}, b_{0,n-1})$  is obtained by:

$$b_{0,j=(n-k,n-k+1,\dots,n-1)} = \begin{cases} 0, & \text{if } \tilde{y}_j < 0 \\ 1, & \text{if } \tilde{y}_j \geq 0 \end{cases} \quad (3.22)$$

Then, the parity-check sub-branch  $B'_0(b_{0,0}, b_{0,n-k-1})$  can be derived from the ordered generator matrix  $\tilde{\mathbf{G}}$ . Concatenate these two sub-branches into the

first complete branch  $B_0$  as follows:

$$B_0 = B'_0(b_{0,0}, b_{0,n-k-1}) \circ B'_0(b_{0,n-k}, b_{0,n-1})$$

Also,

$$\begin{cases} th_{cst} = cst(B_0) \\ \delta(\text{Mag}(\mathbf{y}), cst(B_0)) \geq 0 \end{cases}$$

**Step 2** Applying the growing-forwards process  $V^+$  at the  $(n - k)^{th}$  layer, which corresponds to the less reliable bit in the MRB part, the new branch  $B_1$  is obtained by:

$$B_1 = B_0 \oplus B_0^{en}$$

And the cost threshold is updated as:

$$th_{cst} = \begin{cases} th_{cst}, & \text{if } th_{cst} > cst(B_i) \\ cst(B_i), & \text{if } th_{cst} \leq cst(B_i) \end{cases} \quad (3.23)$$

where  $i = 1$ . If the cost threshold is updated with the new cost, push the current branch into the potential codeword set  $\mathcal{F}$ .

Then the growing direction is determined by the value of  $\delta(\text{Mag}(\mathbf{y}), cst(B_1))$ :

- if  $\delta(\text{Mag}(\mathbf{y}), cst(B_1)) \geq 2|\tilde{y}_{k-2}|$ , processes *Step 3*;
- if  $\delta(\text{Mag}(\mathbf{y}), cst(B_1)) < 2|\tilde{y}_{k-2}|$ , stops and processes *Step 5*.

**Step 3**  $V^+$  process At the  $j^{th}$  layer, the new branch  $B_q$  is obtained by:

$$B_q = B_{q-1} \oplus B_{n-k-j}^{en} \quad (3.24)$$

And update the cost threshold as described in (3.23) with  $i = q$ . If the cost threshold is updated with the new cost, push  $B_q$  into the potential codeword set  $\mathcal{F}$ . Then the direction threshold is also updated as  $\delta(\text{Mag}(\mathbf{y}), cst(B_q))$ ,

- if  $\delta(\text{Mag}(\mathbf{y}), cst(B_q)) \geq 2|\tilde{y}_{k-j-1}|$ ,  $q = q + 1$  and continues *Step 3*;
- if  $\delta(\text{Mag}(\mathbf{y}), cst(B_q)) < 2|\tilde{y}_{k-j-1}|$ , processes  $V^-$  in *Step 4*.

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

*Step 4*  $V^-$  process If the direction threshold indicates a growing-backwards process at the  $j^{th}$  layer, the branch  $B_{q_1}$  is obtained as the same as in (3.24), which is equivalent to cancelling the process of *Step 3* at the  $j^{th}$  vertex, and therefore  $B_{q_1} = B_{q-1}$ , update the cost threshold and the direction threshold as *Step 3*.

- if  $\delta(\text{Mag}(\mathbf{y}), \text{cst}(B_{q-1})) \geq 2|\tilde{y}_{k-j-2}|$ ,  $q = q + 1$  and continues *Step 3*;
- if  $\delta(\text{Mag}(\mathbf{y}), \text{cst}(B_{q-1})) < 2|\tilde{y}_{k-j-2}|$ , processes  $V^-$  in *Step 4*.

*Step 5* When the process stated above finishes at  $v_{ct}$ , evaluate the potential branches in  $\mathcal{F}$  and the one with the most cost is identified as a valid codeword. And a BESCT process is completed.

## 3.5 Numerical Results and Discussion

### 3.5.1 Performance of the BESCT algorithm

We apply the well-know union bound to evaluate the performance of the BESCT algorithm in this section.

$$P_{block} = \sum_{wt=d_{\min}}^n E(wt)Q\left(\sqrt{\frac{2\mathcal{R}\mathcal{E}_b \cdot wt}{\mathcal{N}_0}}\right) \quad (3.25)$$

where  $wt$  is the code weight,  $E(\cdot)$  is the weight enumerator,  $Q(\cdot)$  is the  $Q$ -function<sup>†</sup> and  $\mathcal{R}$  is the coderate.

In 2009, Ambroze *et al.* [2] developed the Tree-search based Codeword Set Enumeration (TCSE) algorithm to search all the codeword sets up to size of threshold  $\tau$  for any parity-check codes. The computational complexity increases as the value of  $\tau$  increasing. By setting a reasonable size of  $\tau$ , a partial weight enumerator table can be achieved. Implementing the union bound  $P_{block}$  in (3.25) with the partial weight enumerator table, a loose-lower union bound  $P_{block}^L$  can

---

<sup>†</sup> $Q(x)$  is the probability that a standard normal random variable will obtain a value larger than  $x$ .

be derived as follows:

$$P_{block}^L = \sum_{wt=d_{min}}^{\tau} E(wt)Q\left(\sqrt{\frac{2\mathcal{R}\mathcal{E}_b \cdot wt}{N_0}}\right) \leq P_{block}$$

The partial weight enumerator table of the PEG-LDPC [32] code (256, 128, 17) has been given in Table 3.1 by setting  $\tau = 20$ .

the PEG-LDPC code (256, 128, 17)				
Code Weight	17	18	19	20
Enumerator	4	13	39	74

Table 3.1: Partial Weight Enumerators Estimated by the TCSE Algorithm [2] (A)

Figure 3.4 illustrates the comparison between the proposed BESCT algorithm and the OSD- $i$  algorithm. The hard-decision decoding (HDD) algorithm is equivalent to the OSD-0 algorithm. The PEG-LDPC code (256, 128, 17) with its minimum Hamming distance of 17 is one of the high  $d_{min}$  LDPC code. As shown in Figure 3.4, the BESCT decoder is capable of performing asymptotically towards to the partial-lower union bound  $P_{block}^L$ . It is also shown that when  $FER < 10^{-4}$ , the performance of the BESCT algorithm has significantly gained more than 2 dB than that of the OSD-3 algorithm, more than 3 dB than that of the OSD-2 algorithm and around 4 dB than that of the OSD-1 algorithm.



### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

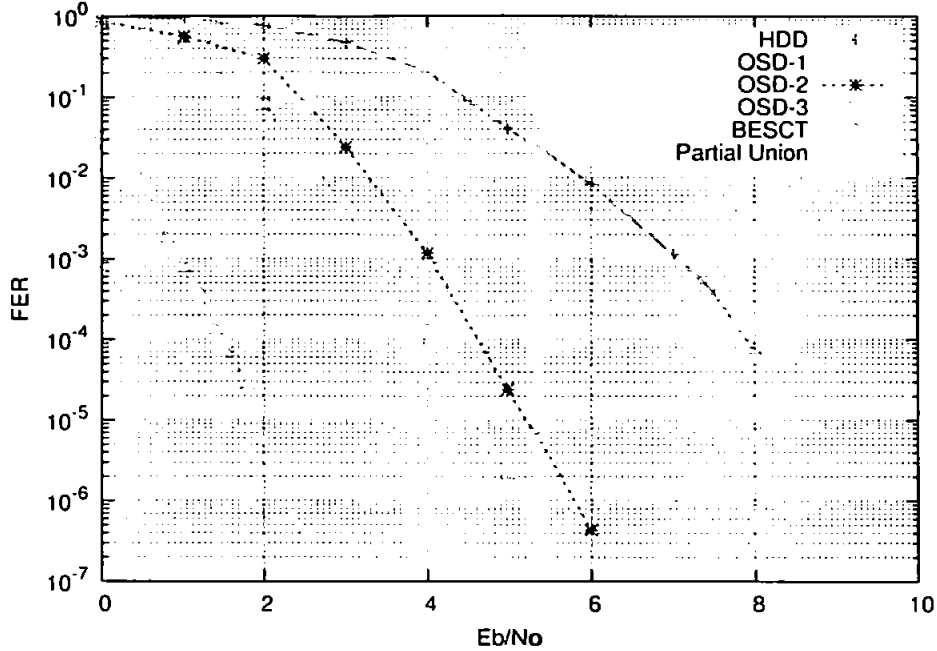


Figure 3.4: Performance of the PEG-LDPC code (256, 128, 17)

Tjhai *et al.* [73] proposed the Euclidean Geometry (EG) LDPC code (63, 37, 9) based on cyclotomic idempotents. This code has been generated with the  $d_{\min}$  of 9 by a sparse  $\mathbf{H}$  matrix. By employing the TCSE weight enumerator search algorithm, the partial weight enumerator table of the EG-LDPC code (63, 37, 9) is given in Table 3.2. As illustrated in Figure 3.5, the OSD-3 is capable of

the EG-LDPC code (63, 37, 9)				
Code Weight	7	8	9	10
Enumerator	0	0	1960	10584

Table 3.2: Partial Weight Enumerators Estimated by the TCSE Algorithm [2] (B)

achieving an MLD performance as the BESCT algorithm. It is also shown that the OSD-2 decoder performs sub-optimal when  $FER > 10^{-5}$ , which means if the transmission condition is defined as  $\frac{E_b}{N_0} \leq 5dB$ , the OSD-2 decoder can be considered as a practical decoder with lower computational complexity than the OSD-3 decoder.

### 3.5 Numerical Results and Discussion

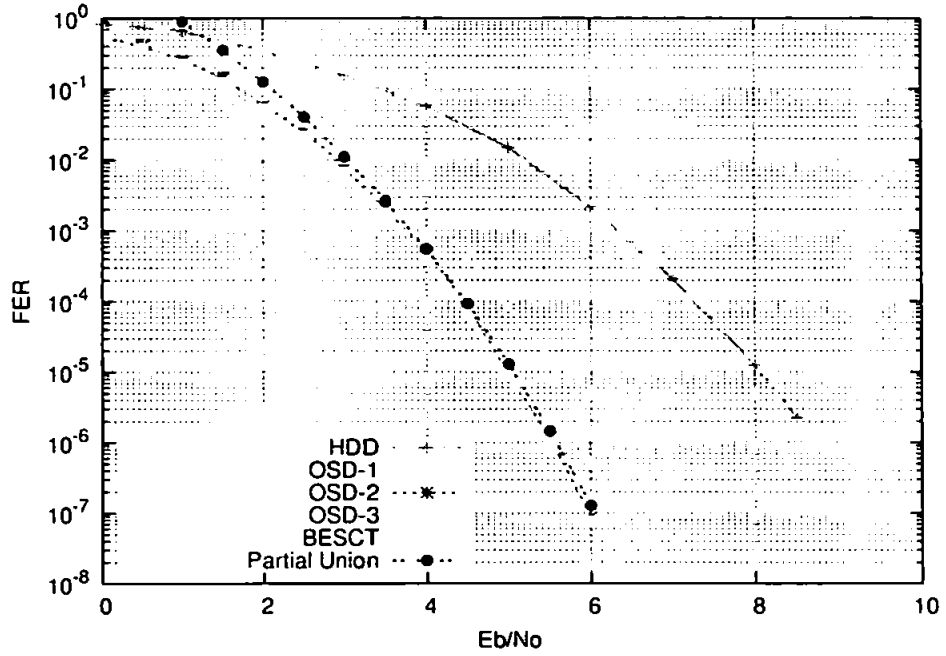


Figure 3.5: Performance of the EG-LDPC code (63, 37, 9)

We also evaluate the performance of the OSML code (255, 175) which has been discussed in Section 2.5. Figure 3.6 depicts that when  $FER < 10^{-1}$ , the performance of the BESCT algorithm has significantly gained more than 1 dB than that of the OSD-3 algorithm, more than 2 dB than that of the OSD-2 algorithm and around 3 dB than that of the OSD-1 algorithm.

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

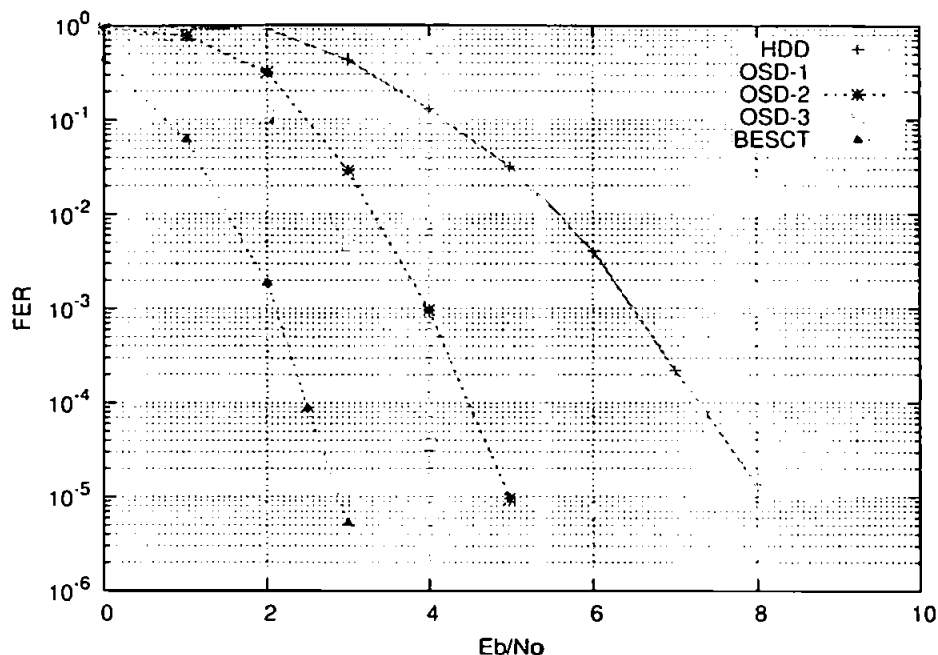


Figure 3.6: Performance of the OSML code (255, 175)

Define  $p$  as a prime congruent to  $+1$  or  $-1 \pmod{8}$ , and denote  $R_0$  as the set of nonzero quadratic residues  $\pmod{p}$ . Let  $\alpha$  be a primitive  $p^{\text{th}}$  root of unity in an extension field of  $GF(2)$ , and let the polynomial  $h(x)$  be defined by  $h(x) = \prod_{r \in R_0} (x - \alpha^r)$ , in which  $h(x)$  is a polynomial with coefficients in  $GF(2)$ . The binary cyclic code of length  $p$  with check polynomial  $h(x)$  is called the *expurgated quadratic residue* (EQR) code, and the code with check polynomial  $(x+1)h(x)$  is called the *augmented quadratic residue* (AQR) code.

The *extended quadratic residue* (LQR) code [4] is defined to be the set of binary  $n$ -vectors of the form  $(c_0, c_1, \dots, c_{p-1}, c_\infty)$ , where  $(c_0, c_1, \dots, c_{p-1})$  is a codeword in the AQR code, and  $c_0 + c_1 + \dots + c_{p-1} + c_\infty = 0$ . It has been known for its powerful error-correcting capability but the difficult analysis of its weight spectrum.

A binary *self-dual* code  $\mathcal{C}$  with its length of  $n$  is a code over  $\mathbb{F}_2$  satisfying  $\mathcal{C} = \mathcal{C}^\perp$ , where  $\mathcal{C}^\perp$  is the dual code of  $\mathcal{C}$  and defined as  $\mathcal{C}^\perp = \{a \in \mathbb{F}_2^n \mid \sum_{i=0}^n a_i b_i = 0 \pmod{2}, \forall b \in \mathcal{C}\}$ . A self-dual code  $\mathcal{C}$  is *doubly even* if all codewords of  $\mathcal{C}$  have their weights divisible by 4, and *single even* if there is at least one codeword with

### 3.5.2 Discussion on the Computational Complexity of the BESCT Algorithm

In this section, we have a discussion on the computational complexity of the BESCT algorithm compared with the OSD algorithm. It has been known that the decoding complexity of the OSD algorithm is determined by the number of potential codewords for the final decision. Suppose the number of orders as  $t$ , the number of potential codewords for the final decision is  $Num(OSD_t) = \sum_{j=0}^t \binom{k}{j}$ .

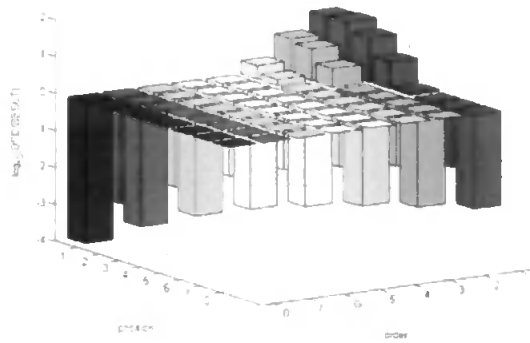
For the BESCT algorithm, the computational complexity of the BESCT algorithm is determined by the number of branches being evaluated during a BESCT process. It is determined by the cost threshold on the  $V^+$  process. Assuming that the cost threshold of the  $V^+$  process is bounded by the vertex at the position of  $n - k + l$ , the number of branches being evaluated is

$$Num(BESCT_l) = \sum_{j=0}^l \binom{l}{j} \tag{3.28}$$

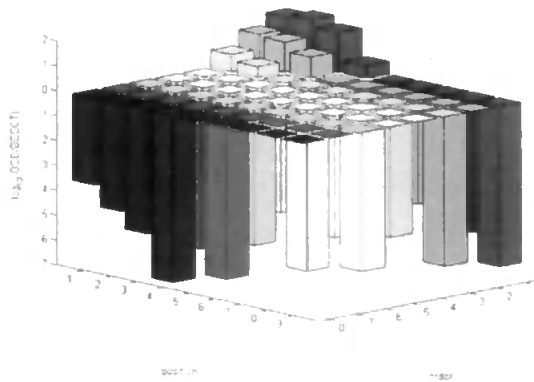
Figure 3.8 from (a) to (e) illustrates the comparison between the OSD algorithm and the BESCT algorithm. It is shown that when the information length is fixed, when  $l \leq 4$  and  $t \leq 4$ , the BESCT algorithm has the computational complexity less than the OSD algorithm.

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

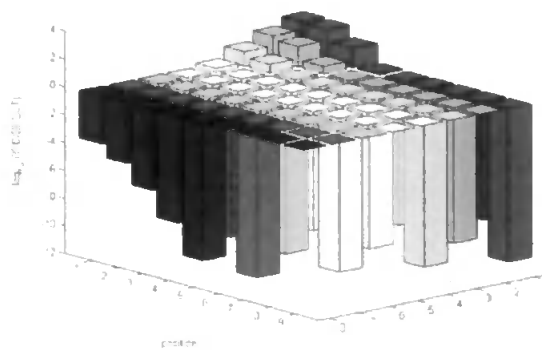
---



(a)  $k = 10$



(b)  $k = 15$



(c)  $k = 20$

### 3.5 Numerical Results and Discussion

---

its weight  $\equiv 2 \pmod{4}$ . A self-dual code is called *extremal* if its  $d_{\min}$  is the highest possible for the given parameters.

The weight enumerator of an extremal doubly-even self-dual code of length  $n$  is given by Gleason's theorem, which has been derived in [56]. Here, we evaluate the BESCT algorithm on the LQR code  $(104, 52, 20)$  which is an extremal doubly-even self-dual code.

$$A(x) = \sum_{i=0}^{\lfloor n/24 \rfloor} E(i)(1 + 14x^4 + x^8)^{\frac{n}{8}-3i} \{x^4(1-x^4)^4\} \quad (3.26)$$

By (3.26), the weight spectrum of the LQR code  $(104, 52, 20)$   $A_{\text{LQR}(104,52,20)}$  is obtained as follows:

$$\begin{aligned} A_{\text{LQR}(104,52,20)}(x) = & (x^0 + x^{104}) + \\ & 1138150 \cdot (x^{20} + x^{84}) + \\ & 206232780 \cdot (x^{24} + x^{80}) + \\ & 15909698064 \cdot (x^{28} + x^{76}) + \\ & 567725836990 \cdot (x^{32} + x^{68}) + \\ & 9915185041320 \cdot (x^{40} + x^{64}) + \\ & 88355709788905 \cdot (x^{44} + x^{60}) + \\ & 413543821457520 \cdot (x^{48} + x^{56}) + \\ & 1406044530294756 \cdot (x^{52}) \end{aligned} \quad (3.27)$$

As shown in Figure 3.7, when  $FER < 10^{-4}$ , the code performance of the BESCT algorithm has been improved by 0.5 dB than that of the OSD-3 algorithm, 1 dB than that of the OSD-2 algorithm and around 5 dB than that of the OSD-1 algorithm.

### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

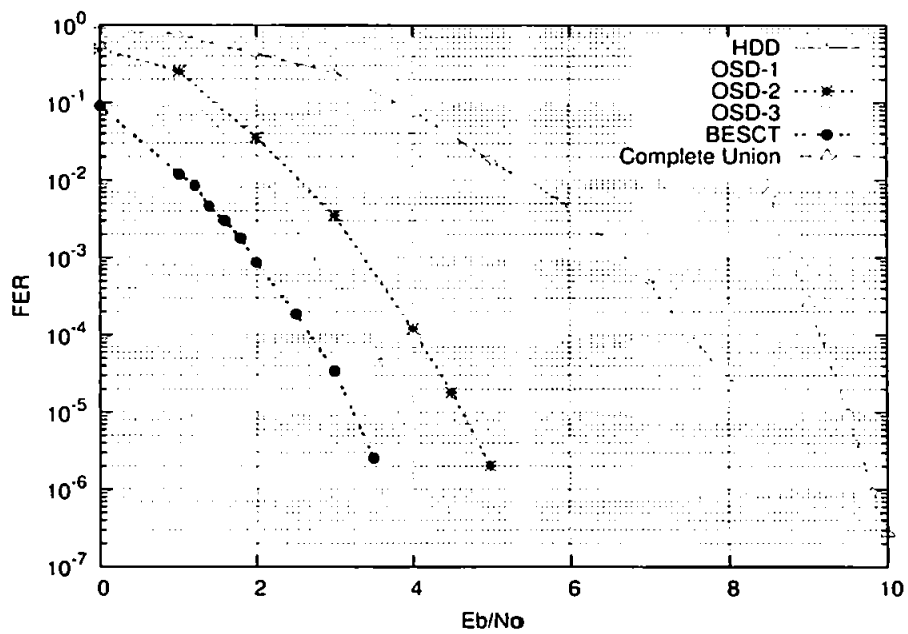
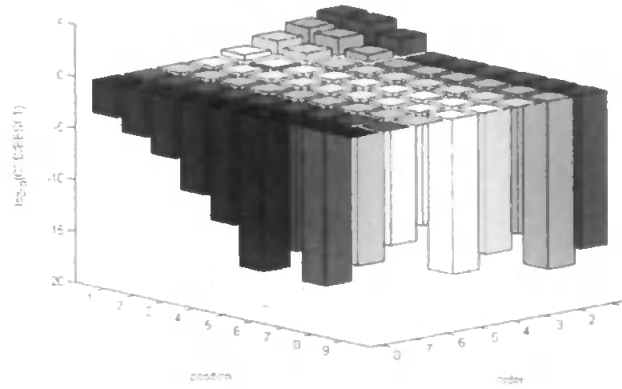
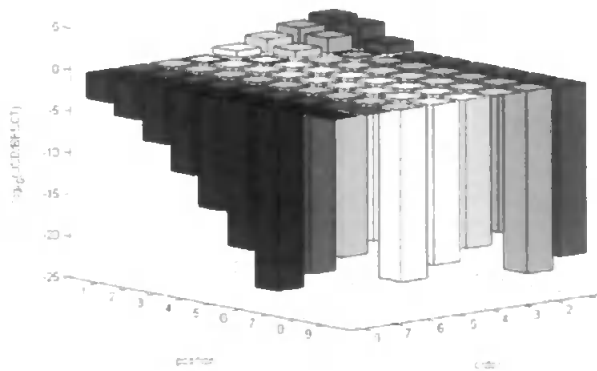


Figure 3.7: the Frame-error-rate performance of the LQR (104, 52, 20)

### 3.5 Numerical Results and Discussion



(d)  $k = 25$



(e)  $k = 30$

Figure 3.8: The Complexity Comparison between the OSD algorithm and the BESCT algorithm



### 3. BRANCH-EVALUATION SEARCH ON THE CODE-TREE ALGORITHM

---

#### 3.6 Summary

In this chapter, a graph-based ML-approaching algorithm for the AWGN channel is proposed. By applying a bi-directional code tree, the proposed branch-evaluation algorithm based on the cross-correlation metric is equivalent to the classic Dorsch algorithm (or the OSD algorithm with the order of  $k$ ) with a flexible threshold method. The dynamic threshold method speeds up the decoding process when the number of errors in the MRB is less than 4, the position of the least likable error is located within the first 4 less reliable bits in the MRB, compared to the OSD algorithm.

## Part II

# Packet Data Transmission and Coding Arrangement

*When I have fully decided a result is worth getting I go ahead of it and make trial after trial until it comes.*

Thomas A. Edison

# 4

## The Packet Data Transmission System of Fountain Codes

### 4.1 Background

Fountain codes [38, 41] have been claimed as a class of sub-optimal erasure codes with the property that a potential limitless sequence of transmitted packets  $\{y_0, y_1, y_2, \dots\}$  can be generated from a given set of  $k$  information packets  $\{x_0, x_1, \dots, x_{k-1}\}$  such that the information packets can ideally be recovered from any subset of the transmitted packets of size  $k'$ , which is slightly greater than the number of information packets, that is,  $k' \geq k$ .

The first practical realisation of fountain codes is so-called “*Luby-Transform*” (LT) codes which was invented in 2002 by Luby [38]. The concept of the “*rateless*” transmission is introduced in [38], which means the encoding algorithm is capable of in principle producing an arbitrarily large number of packets which can be transmitted until the receivers recover the original information packets.

## 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

One of the advantages of LT codes is the low computational complexity on their encoding and decoding algorithms by employing the *exclusive-OR* (XOR) operations on the packets. The XOR algorithm is equivalent to the recovery algorithm applied for other linear block codes over the BEC. However, LT codes suffer from an error-floor problem due to the existence of the stopping sets which is caused by their code structures.

To improve the performance, especially the error-floor caused by the nature of LT codes, in 2003, Shokrollahi [68] introduced a class of two-layer fountain codes, called as “*Raptor*” codes. Raptor codes maintain the “rateless” property since their inner codes are constructed based on LT codes, and improves the performance of LT codes by applying outer codes to enhance the ability of the erasure/error correction. As the class of the most effective fountain codes, Raptor codes have been deployed in the 3GPP MBMS standard for broadcasting file delivery and streaming services, in the DVB-H IPDC standard for delivering IP services over DVB networks, and DVB-IPTV for delivering commercial TV services over an IP network. [78]

The good efficiency and performance of fountain codes motivate us to study the code structure and the application of them. We study, in Section 4.2, the encoding process and the decoding process of LT codes with a proposed matrix representation. In Section 4.2.3, we investigate the degree distribution for the construction of LT codes, which determines the performance of LT codes. Section 4.3 study Raptor codes. We also propose a matrix representation for the construction of Raptor codes in Section 4.3. In Section 4.4, we provide the simulation results and discussion on the limitation of fountain codes. We then conclude this chapter and give some insights about the results in Section 4.5

### 4.2 LT Codes

It is useful to know the definitions of the *Degree* and the *Degree Distribution* before the study of fountain codes.

**4.1 Definition (Degree).** The degree of a vertex  $v$  is the number of edges that are incident with vertex  $v$ , i.e. the number of edges that are connected to vertex  $v$ , denoted as  $d$ .

**4.2 Definition (Degree Distribution).** Designate  $\rho(d)$  as a distribution function of  $d$ . For all  $d$ ,  $\rho(d)$  is the probability that an encoding symbol has degree  $d$ .

LT codes are controlled by the parameters as listed as follows:

- the number of information packets, denoted as  $k$
- the degree distribution  $\rho(d)$

Thus, an LT code is specified as  $\mathcal{C}_{\text{LT}}(k, \rho(d))$  and its encoding process is a linear map  $\mathbb{F}_2^k \rightarrow \mathbb{F}_2^n$ .

### 4.2.1 LT Encoding Process

Suppose the information packets as  $\mathbf{x} = (x_1, \dots, x_k)$ , and then each transmission packet  $u_j$  is generated by

1. randomly choosing the degree  $d_i$  from a given degree distribution  $\rho(d)$ ;
2. randomly choosing  $d_i$  distinct information packets and exclusive-oring those  $d_i$  information packets to obtain a transmission packet  $u_j$ .

Algorithm 4.1 gives the pseudo code of LT encoding process.

Figure 4.1 shows an example of an LT encoding process by defining a bipartite graph which connects transmission packets to information packets.

Alternatively, LT codes can also be represented in a matrix form  $\mathbf{M}^{\text{LT}}$  which only contains the non-systematic part of its parity-check matrix  $\mathbf{H}$ .

$$\mathbf{M}^{\text{LT}} = \begin{pmatrix} m_{0,1}^u & m_{0,2}^u & \cdots & m_{0,k}^u \\ m_{1,1}^u & m_{1,2}^u & \cdots & m_{1,k}^u \\ \vdots & \ddots & \ddots & \vdots \\ m_{k-1,1}^u & m_{k-1,2}^u & \cdots & m_{k-1,k}^u \\ \vdots & \ddots & \ddots & \vdots \end{pmatrix} \quad (4.1)$$

## 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---



---

### Algorithm 4.1 $LT_{Enc}(\rho(\cdot), \mathbf{x}, \mathbf{u})$

---

**Input:**

- $\rho(\cdot) \Leftarrow$  Degree Distribution
- $\mathbf{x} \Leftarrow$  Information packets
- $\mathbf{y} \Leftarrow$  Transmission packets
- $rand.int(a, b)$  is to randomly choose an integer from  $a$  to  $b$
- $rand.float(a, b)$  is to randomly choose a floating data from  $a$  to  $b$
- $\rho(\cdot)$  is a value-increasing array
- $\omega$  is a floating data satisfying  $0 \leq \omega \leq 1$

**Output:**

- 1: **repeat**
  - 2:    $\omega = rand.float(0, 1)$
  - 3:   **if** ( $\omega < \rho(d_j)$ ) **then**
  - 4:      $d_j$  is the chosen degree
  - 5:   **end if**
  - 6:   choose uniformly at random  $d$  information packets  $x_{i_1}, \dots, x_{i_d}$
  - 7:    $u_j = x_{i_1} \oplus x_{i_2} \oplus \dots \oplus x_{i_d}$
  - 8: **until** ( $j$  satisfies the number of transmission packets)
- 

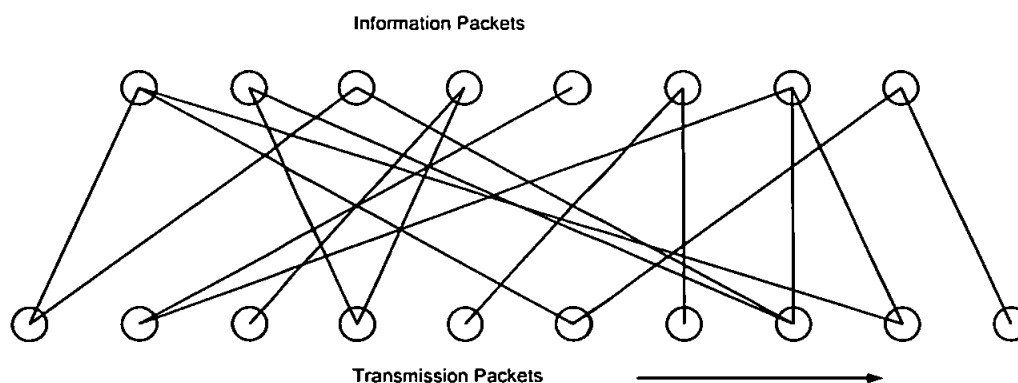


Figure 4.1: LT Encoding Processing

Henceforth, the LT code graphically represented in Figure 4.1 can also be represented by the  $M$  matrix representation as given as follows:

$$M = \begin{pmatrix} 10000000 \\ 01000001 \\ 01001100 \\ 00100000 \\ 10000001 \\ 00100000 \\ 00001010 \\ 00001000 \\ 01010000 \\ 00000101 \end{pmatrix}$$

### 4.2.2 LT Decoding Process

Designate  $\mathbf{y} = \{y_0, y_1, \dots\}$  as the received packets. Each received packet  $y_i$  should include:

- the information of degree  $d_i$  applied for this packet
- the index of information packets which are connected to this packet.

The decoding process starts when the receiver collects  $k'$  packets. The number of packets  $k'$  is required to be at least equal to  $k$ , that is,

$$k' = k + \delta \quad (4.2)$$

where  $\delta$  is a small positive number.

In [38], the proposed decoding algorithm is called as “XOR” algorithm. Designate the recovered packets as  $s_i$ , and the XOR algorithm is described as follows:

- 1.) Start with a received packet  $y_j$  containing the degree  $d_j = 1$  which means that the received packet  $y_j$  is only connected to an information packet  $s_i$ .
  - 1.1) Set  $s_i = y_j$
  - 1.2) Add  $s_i$  to all received packets  $y_t$  that are connected to  $s_i$ .

## 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

1.3) Remove the index of  $s_i$  from the header of the received packets  $y_t$  and  
 $d_t = d_t - 1$ .

2.) Repeat 1.) until all information packets are recovered.

The XOR decoder stops when either no output symbols of degree one, that is  $\{d_j = 1 | \forall j \in \{1, 2, \dots, k'\}\}$ , or all the information packets have been recovered.

The LT decoding algorithm is given in Algorithm 4.2.

---

### Algorithm 4.2 LTDec-XOR( $y, \rho(d), s$ )

---

**Input:**

$y \leftarrow$  received packets  
 $\rho(d) \leftarrow$  Degree Distribution  
 $Deg(\cdot) \leftarrow$  Degree Reader from the packet header  
 $i \leftarrow$  Received packet index  
 $j \leftarrow$  Recovered information packet index  
**Buf**  $\leftarrow$  buffer

**Output:**

$s \leftarrow$  successfully recovered information packets

- 1: **repeat**
- 2:   **if** ( $Deg(y_i) \neq 1$ ) **then**
- 3:     push  $y_i$  in **Buf**
- 4:   **end if**
- 5:   **if** ( $Deg(y_i) == 1$ ) **then**
- 6:      $s_j \Rightarrow y_i$
- 7:   **end if**
- 8:   **for all**  $y_i \in$  **Buf**:  $y_i$  includes  $s_j$  **do**
- 9:      $y_i \Rightarrow y_i \oplus s_j$
- 10:  **end for**
- 11: **until** ( original information packets are recovered)

---

### 4.2.3 Degree Distribution $\rho(d)$

It has been known that the degree distribution of an LT code is of a great importance.

- The majority of the transmission packets must have low degree, so that the decoding process can get started and then keep going.



- Occasionally, the information packets must be encoded with high degrees to avoid the null connection between the information packets and the transmission packets.

The ideal degree distribution is approached by the *Ideal Soliton Distribution*.

$$\rho(d) = \begin{cases} \frac{1}{k}, & \text{if } d = 1 \\ \frac{1}{d(d-1)}, & \text{for } d = 2, 3, \dots, k. \end{cases} \quad (4.3)$$

Designate  $\Omega(x)$  as the degree distribution function as follows:

$$\Omega(x) = \sum_{i=1}^k \rho(d_i)x^{d_i} \quad (4.4)$$

Assume  $k = 15$  and then the degree distribution function can be derived from (4.3) as follows:

$$\begin{aligned} \Omega(x) = & 0.066667x^1 + 0.5x^2 + 0.166667x^3 + 0.083333x^4 + 0.05x^5 \\ & + 0.033333x^6 + 0.023810x^7 + 0.017857x^8 + 0.013889x^9 + 0.011111x^{10} \\ & + 0.009091x^{11} + 0.007576x^{12} + 0.006410x^{13} + 0.005495x^{14} + 0.004762x^{15} \end{aligned}$$

However, the Soliton distribution does not work as good as expected mainly because a few information packets have null-connections with the transmission packets.

As shown in Figure 4.2, the LT encoding process with the ideal Soliton distribution, has a very trivial probability to constitute a transmission packet from a single information packet. As the consequence of it, at receiver, the decoding process is hardly to be implemented because of no degree-one received packet.

Following the analysis of Luby [38], the problem of the ideal Soliton distribution is that the expected *ripple*\* size (one) is too small.

Henceforth, Luby [38] proposed a robust Soliton distribution, which normalised the ideal Soliton distribution with an expected distribution. In his anal-

---

\*The set of covered input symbols that have not yet been processed is called the ripple, and thus at this point all covered input symbols are in the ripple.

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

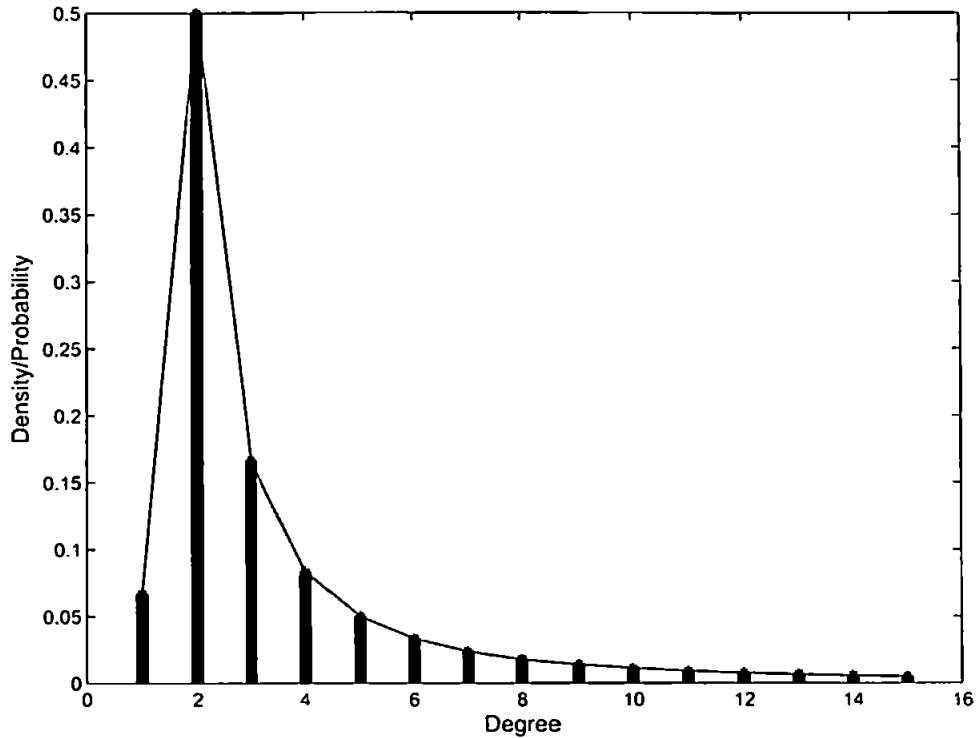


Figure 4.2: the Degree Distribution Generated from the Ideal Soliton Distribution for a LT code with information length of 15

ysis, the robust Soliton distribution ensures every information packet is likely to be connected to a transmission packet at least once, so that the decoding process gets initialised and keeps going.

In our analysis, we applied the Shokrollahi [69] degree distributions which have been also applied in the IEFT standard.

Table 4.1 shows the optimised degree distributions with  $k = 65536, 80000, 100000, 120000$ , under the condition of  $\delta = 0.01$ .

$k$	65536	80000	100000	120000
$\Omega_1$	0.007969	0.007544	0.006495	0.004807
$\Omega_2$	0.493570	0.493610	0.495044	0.496472
$\Omega_3$	0.166220	0.166458	0.168010	0.166912
$\Omega_4$	0.072646	0.071243	0.067900	0.073374
$\Omega_5$	0.082558	0.084913	0.089209	0.082206
$\Omega_8$	0.056058		0.041731	0.057471
$\Omega_9$	0.037229	0.043365	0.050162	0.035951
$\Omega_{18}$				0.001167
$\Omega_{19}$	0.0055590	0.045231	0.038837	0.054305
$\Omega_{20}$		0.010157	0.015537	
$\Omega_{65}$	0.025023			0.018235
$\Omega_{66}$	0.003135	0.010479	0.016298	0.009100
$\Omega_{67}$		0.017365	0.010777	

Table 4.1: Shokrollahi Degree Distribution [69]

Figure 4.3 illustrates a comparison of the degree distributions between the ideal Soliton distribution and the Shokrollahi distribution. As can be seen, even when the number of information packets is as small as 1024, the Shokrollahi method can produce some transmission packets with their degrees of one, while the ideal Soliton distribution generates null. Meanwhile, the Shokrollahi method maintains desirable proportions of the generation of degree-2 and degree-3 transmission packets.

## 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

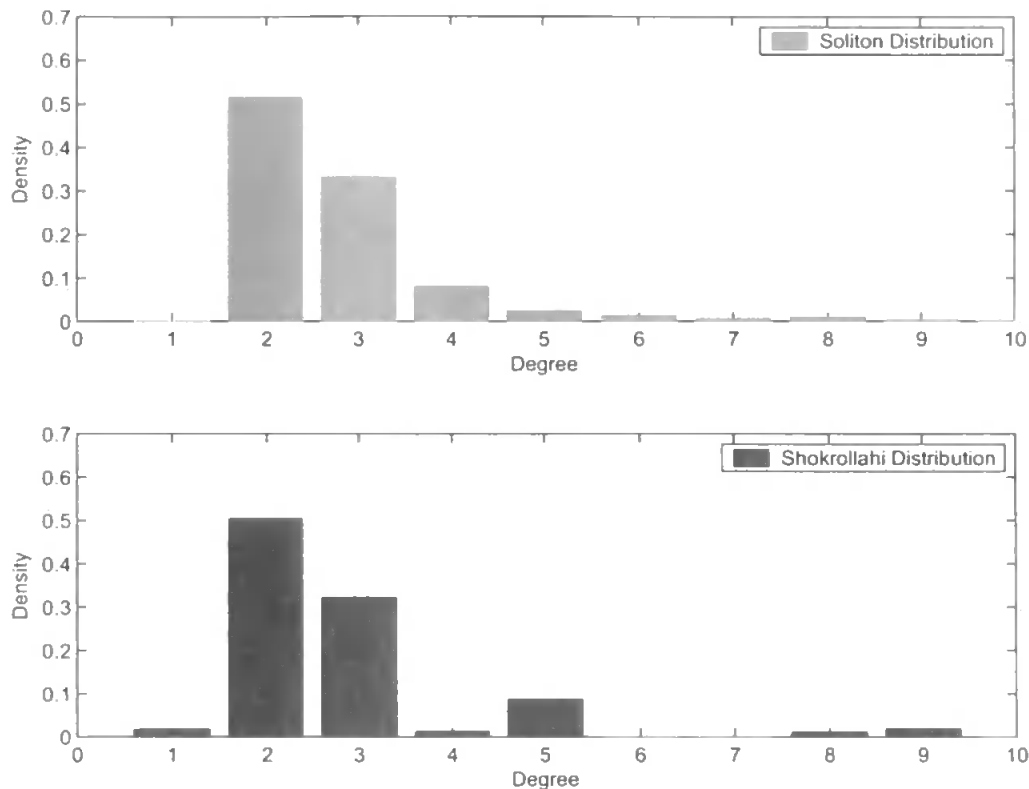


Figure 4.3: Degree Distributions based on Different Methods,  $Pkt_{no}(info) = 1024$

### 4.3 Raptor Codes

The performance of LT codes is restricted by the stopping sets due to its sparse structure. When the network condition is poor, an error floor will result in a lot of packet loss and huge demanding on re-sending packets. To solve this problem, Shokrollahi [69] invented *Raptor codes*, which maintain the desirable properties of LT codes, such as the capability of generating transmission packets as many as required and the simple recovery process with the XOR decoder, and provide a better performance.

Denote a Raptor code as  $\mathcal{C}^{Rpt}$  and it is formed by concatenating an outer code  $\mathcal{C}^p$  (also called the *pre-code* in terms of the encoding process), with an LT code  $\mathcal{C}^{LT}$ .

4.3 **Definition (Raptor code).** A Raptor code with parameters  $(k, \mathcal{C}^p, \Omega(d))$  is an LT-code with degree distribution function  $\Omega(d)$ , which has given in Table. 4.1 on  $k + s$  pre-coded packets which are coordinates of codewords in  $\mathcal{C}^p$ .

Suppose that the pre-code  $\mathcal{C}^p$  is with its code length of  $k+s$  and its information length of  $k$ . Hence, after the pre-coding process,  $k + s$  intermediate packets are generated by encoding  $k$  information packets with a linear block code  $\mathcal{C}^p(k + s, k)$ , which is designed to decode the unrecovered information packets from the recovered packets after the LT decoding process.

4.4 **Definition (Intermediate packet).** Given  $k$  information packets  $(x_0, x_1, \dots, x_{k-1})$ ,  $k + s$  intermediate packets  $(f_0, f_1, \dots, f_{k+s-1})$  are uniquely defined as

- The information packets satisfy the  $k$  constrains of LT-code  $\mathcal{C}^{LT}$  initialised before the transmission.
- The  $k + s$  intermediate packets satisfy the pre-coding  $\mathcal{C}^p$  relationship initialised before the transmission.

### 4.3.1 Non-Systematic Raptor Codes

A non-systematic Raptor encoding process were first introduced as a graphical expression in [69] as shown in Figure 4.4, which is a two-layered encoding structure.

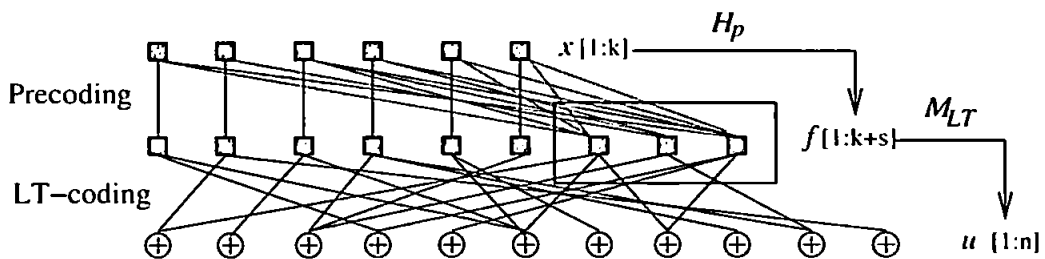


Figure 4.4: the Graphical Expression of a non-systematic Raptor Encoding Process

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

The details in the non-systematic Raptor encoding process are listed as follows:

1. The information packets  $\mathbf{x} = (x_1, x_2, \dots, x_3)$  are pre-encoded into the intermediate packets  $\mathbf{f} = (f_1, f_2, \dots, f_{k+s})$  by a linear block code  $\mathcal{C}_p$ . The linear block code  $\mathcal{C}_p$  is with the information length of  $k$  and the code length of  $k + s$ . As depicted in Figure 4.4, the precoding process employs the row-echelon parity-check matrix  $\mathbf{H}^p$  of the code  $\mathcal{C}_p$ . The  $\mathbf{H}^p$  is defined as:

$$\mathbf{H}^p = \begin{pmatrix} h_{0,1}^p & \cdots & h_{0,k}^p & 1 & 0 & 0 & \cdots \\ h_{1,1}^p & \cdots & h_{1,k}^p & h_{1,k+1}^p & 1 & 0 & \cdots \\ \vdots & \ddots & & \vdots & \ddots & \ddots & \vdots \\ h_{s-1,1}^p & \cdots & h_{s-1,k}^p & \cdots & \cdots & h_{s-1,k+s-1}^p & 1 \end{pmatrix} \quad (4.5)$$

Then, the precoding process is equivalent to:

$$f_j = \begin{cases} x_j, & \text{if } j \leq k \\ \sum_{i=1}^j \oplus f_i h_{j,i}^p, & \text{if } k < j \leq k + s \end{cases} \quad (4.6)$$

where  $\sum_{i=1}^j \oplus a_i b_i = a_1 b_1 \oplus a_2 b_2 \oplus \cdots \oplus a_j b_j$ .

2. By employing the LT encoding Algorithm 4.1, the intermediate packets are further encoded into the transmission packets  $\mathbf{u}$ .

At receiver, the corresponding decoding process for a non-systematic Raptor code also involves two decoders for the inner LT code  $\mathcal{C}^{LT}$  and the outer pre-code  $\mathcal{C}^p$  respectively.

#### 4.3.2 Systematic Raptor Codes

In a communication system, a systematic format code is always preferred. In [40], the algorithm has been proposed to convert a non-systematic Raptor code into a systematic one by applying a restriction on the information packet before the pre-code.

In this section, instead of generating an invertible  $k \times k$  matrix [68] to pre-process the information packets, a simplified method to constitute a systematic Raptor code is given.

First of all, we introduce a concept of the formatting matrix  $\mathbf{A}$ , which is not a strictly traditional  $\mathbf{G}$  due to the rateless property of Raptor codes. The formatting matrix  $\mathbf{A}$  is to concatenate the generator matrix  $\mathbf{G}^p$  of the pre-code  $\mathcal{C}^p$  with the  $\mathbf{M}^{LT}$  of the LT code.

Designate  $\mathbf{A}_1^{Rpt}$  as the *formatting matrix* for the non-systematic Raptor code, which concatenates the matrix  $\mathbf{G}^p$  and the  $\mathbf{M}^{LT}$  directly as given in (4.7):

$$\begin{aligned} \mathbf{A}_1^{Rpt} &= \begin{pmatrix} g_{0,1}^p & \cdots & g_{0,k}^p & 1 & 0 & \cdots & 0 \\ g_{1,1}^p & \cdots & g_{1,k}^p & 0 & 1 & \cdots & 0 \\ \vdots & \ddots & \ddots & \vdots & \ddots & \ddots & \vdots \\ g_{k-1,1}^p & \cdots & g_{k-1,k}^p & 0 & 0 & \cdots & 1 \\ m_{0,1}^{lt} & \cdots & m_{0,k}^{lt} & m_{0,k+1}^{lt} & \cdots & \cdots & m_{0,k+s-1}^{lt} \\ m_{1,1}^{lt} & \cdots & m_{1,k}^{lt} & m_{1,k+1}^{lt} & \cdots & \cdots & m_{1,k+s-1}^{lt} \\ \vdots & \ddots & \ddots & \ddots & \vdots & \vdots & \vdots \\ m_{k+s-1,1}^{lt} & \cdots & m_{k+s-1,k}^{lt} & m_{k+s-1,k+1}^{lt} & \cdots & \cdots & m_{k+s-1,k+s-1}^{lt} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \end{pmatrix} \\ &= \begin{pmatrix} \mathbf{G}^p \\ \mathbf{M}^{LT} \end{pmatrix} \end{aligned} \quad (4.7)$$

The rank of the formatting matrix  $\mathbf{A}_1^{Rpt}$  is determined by the required number of transmission packets, which ensures a successful recovery over the  $k$  information packets.

Alternatively, the encoding process can also be described as follows:

1. Calculate  $k + s$  intermediate packets  $\mathbf{f}$  as:

$$\mathbf{f} = \mathbf{x} \cdot \mathbf{G}^p \quad (4.8)$$

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

2. Calculate the transmission packets  $\mathbf{u}$  as:

$$\mathbf{u} = \mathbf{f} \cdot (\mathbf{M}^{LT} | \mathbf{I}) \quad (4.9)$$

If the  $k + s$  transmission packets are generated by a systematic code, the  $k + s$  columns of  $\mathbf{A}_1^{Rpt}$  should be *independent* to each other. That is, if applying the In-place algorithm on the first  $k + s$  rows of the  $\mathbf{A}_1^{Rpt}$ , a full rank square submatrix  $((k + s) \times (k + s))$  should be obtained.

After processing the In-place algorithm, if the square submatrix is with the rank less than  $k + s$ , the column-swapping process on the first  $k$  columns is required. Demote the swappings as a permutation function as  $\pi(\cdot)$ . And then, the formatting matrix for the modified Raptor code is written as  $\mathbf{A}_{11}^{Rpt}$ :

$$\mathbf{A}_{11}^{Rpt} = \pi(\mathbf{A}_1^{Rpt}) \quad (4.10)$$

The first  $k + s$  rows in  $\mathbf{A}_{11}^{Rpt}$  correspond to the positions of the systematic transmission packets. The systematic transmission packets are generated from the ordered information packets  $\mathbf{x}' = \pi^{-1}(\mathbf{x})$ .

The conceptual block diagram of a systematic Raptor encoder which deals the Raptor code as a concatenation code, is shown in Figure 4.5.

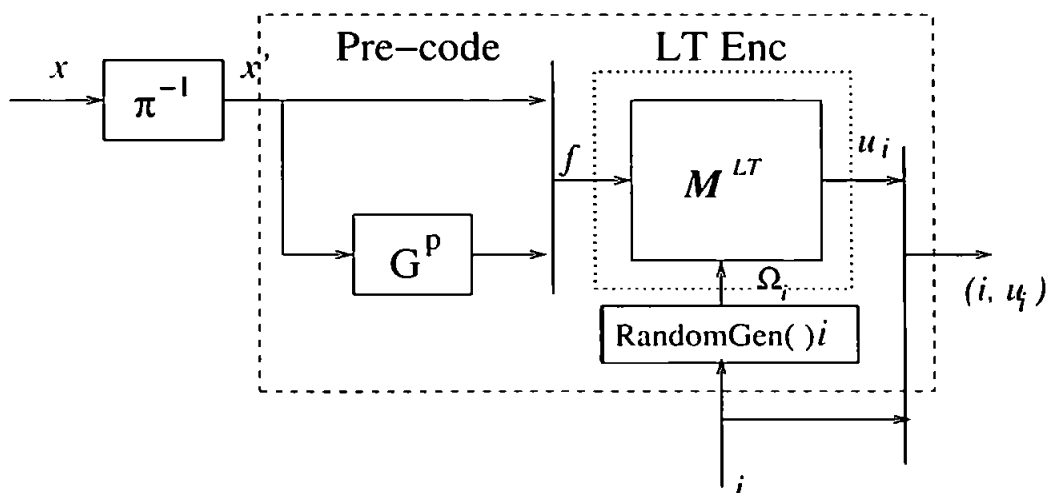


Figure 4.5: Systematic Raptor encoder



At receiver, the corresponding decoder works to re-generate the matrix of  $\mathbf{A}_{\text{II}}^{Rpt}$  with the information contained in the headers of received packets. Once the matrix of  $\mathbf{A}_{\text{II}}^{Rpt}$  is verified to contain a section of  $(k + s) \times (k + s)$  matrix with a full rank. The missed information packets can be re-encoded with this systematic matrix.

## 4.4 Numerical Results and Discussion

In the works of MacKay [41], Shokrollahi [68], LT codes and Raptor codes have been proved as a class of sub-optimal erasure codes. Therefore, the actual number of transmission packets for a successful transmission, as denoted as  $k'$ , and the number of the original information packets, as defined as  $k$  have the relationship of  $k' \approx k$ .

Even though it has been claimed that  $k' \approx k$ , it is still very crucial to know the number of extra transmission packets required in order to recover the whole information file.

Designate  $\delta = k' - k$  as the number of the extra transmission packets to contribute a successful transmission. The value of  $\delta$  is the so-called *overhead* in a complete transmission, which is a measure of the efficiency and the computational complexity in an erasure coding scheme.

The first discussion is on the relationship between the overhead and the size of the information file during a transmission via the BEC. The degree distribution function applied in the simulation is given as follows:

$$\begin{aligned} \Omega(x) = & 0.066667x^1 + 0.5x^2 + 0.166667x^3 + 0.083333x^4 + 0.05x^5 \\ & + 0.033333x^6 + 0.023810x^7 + 0.017857x^8 + 0.013889x^9 + 0.011111x^{10} \\ & + 0.009091x^{11} + 0.007576x^{12} + 0.006410x^{13} + 0.005495x^{14} + 0.004762x^{15} \end{aligned} \quad (4.11)$$

Equation (4.11) has also been employed in the 3GPP [1] as the standard distribution when  $k \leq 2^{16}$ .

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

Figure 4.6 illustrates the histogram of the actual number of required transmission packets to ensure a successful recovery on a file of 10,000 information packets, when the channel erasure probability  $p$  equals 0.5. In other words, the probability of each transmission packet being erased during the transmission is 50%. The mean value of  $k'$  is 10472, that is, for a full recovery of 10,000 information packets, the system requires the transmission overhead of  $4.72\%k$  which is a reasonably small number.

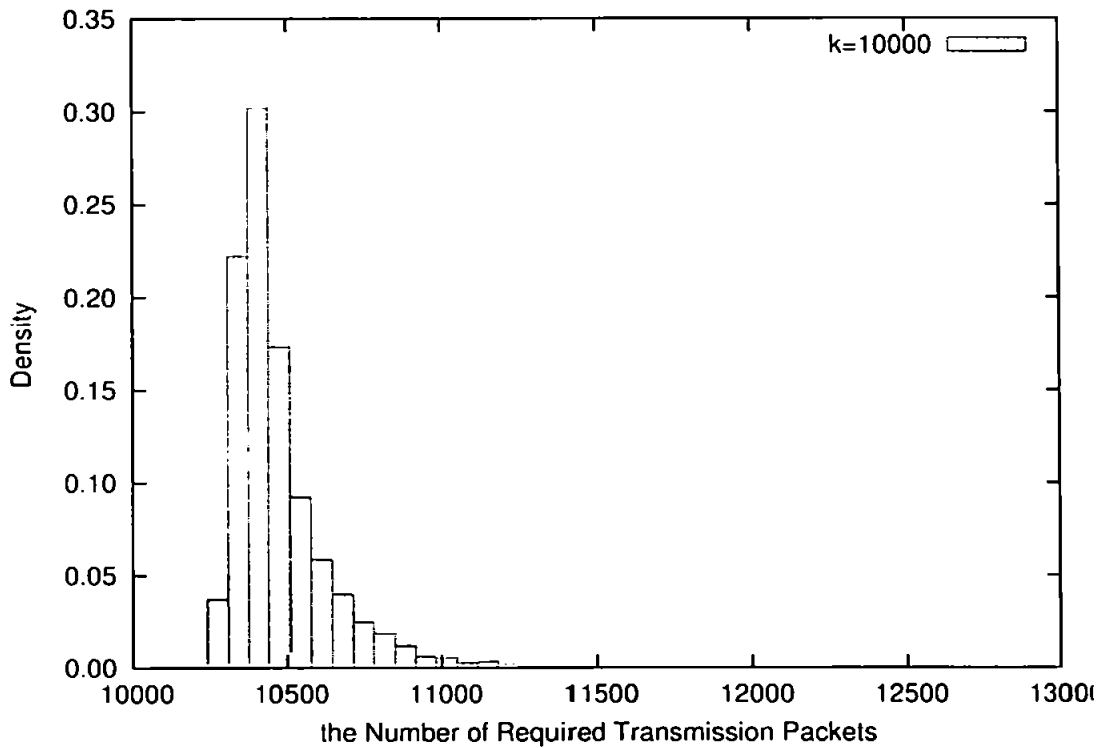


Figure 4.6: Histogram of the actual number of required transmission packets when the number of information packets is 10,000 and  $p = 0.5$

#### 4.4 Numerical Results and Discussion

By halving the number of information packets in Figure 4.6, the LT code generated with the same degree distribution function requires an overhead of a slightly more percentage over the size of the information file. As shown in Figure 4.7, the mean value of the actual number of the transmission packets is 5320 and the overhead percentage is 6.4%.

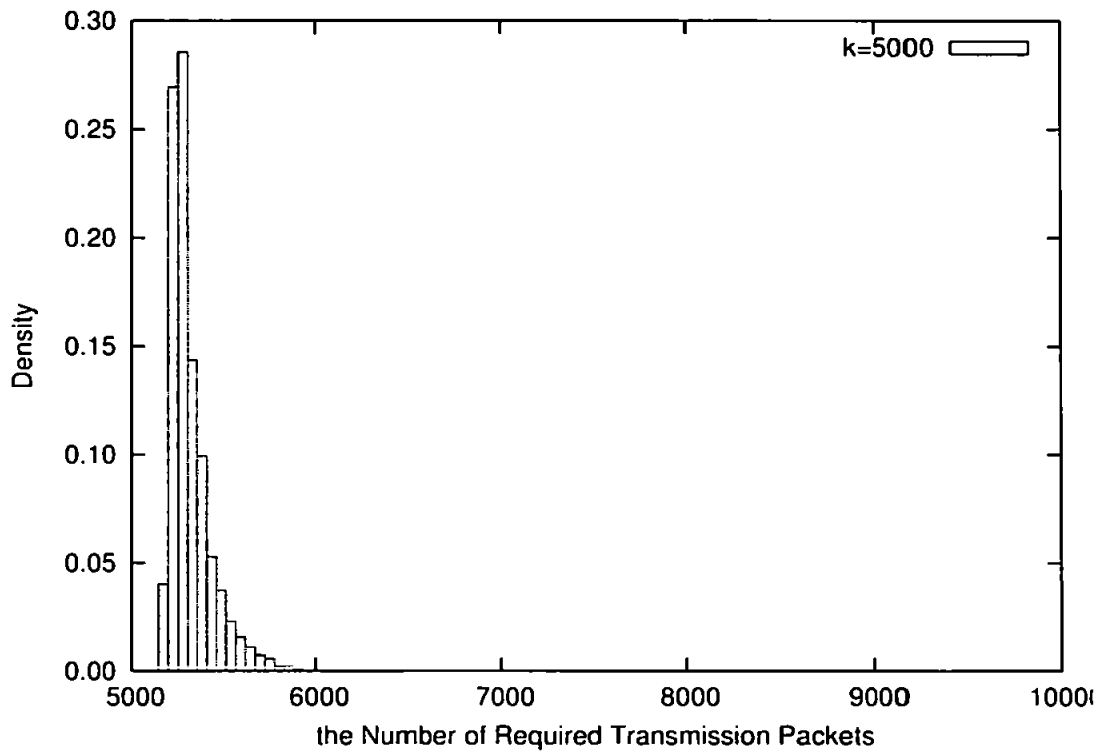


Figure 4.7: Histogram of the actual number of required transmission packets when the number of information packets is 5000 and  $p = 0.5$

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

The same LT code as above has been employed to transmit an information file with 1024 packets. The histogram in Figure 4.8 implies a much higher overhead percentage, which is 12.7% and the mean value of the number of the required transmission packets is 1154.

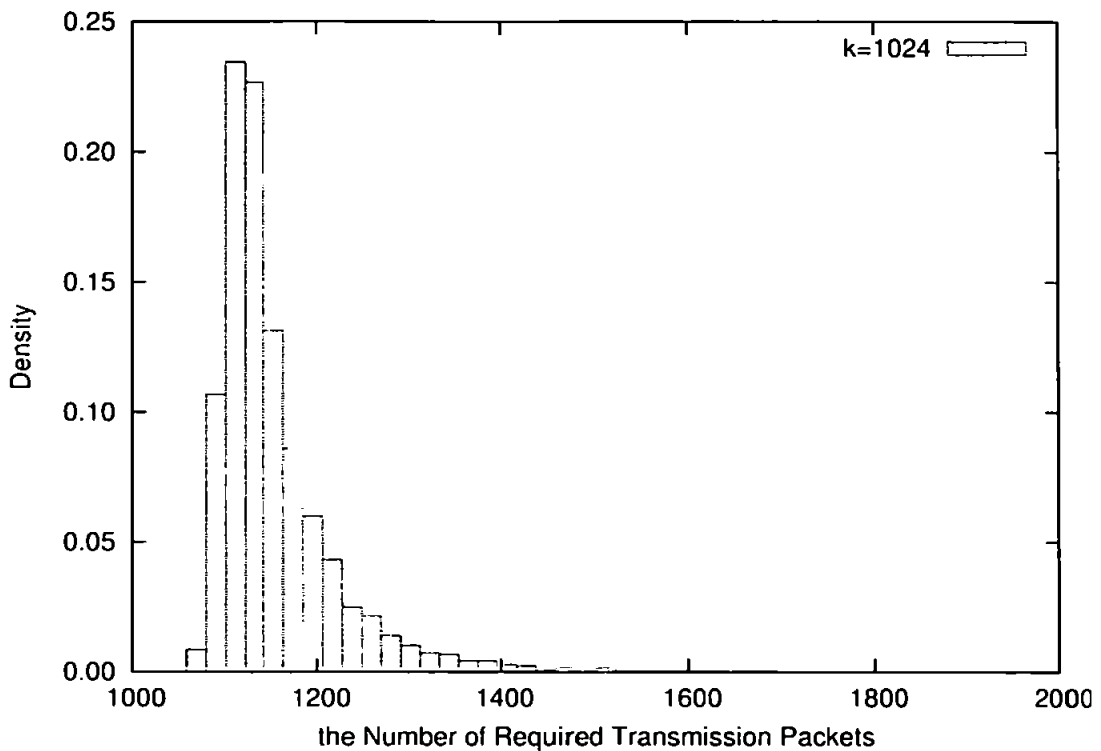


Figure 4.8: Histogram of the actual number of required transmission packets when the number of information packets is 1024 and  $p = 0.5$

Figures 4.9, 4.10 and 4.11 exhibit the histograms under the condition of different sizes of the information files of 512, 250 and 125, respectively. The less the information packets, the higher the overhead percentage is. Especially, when the information file with the packet number of 125, a successful transmission requires an average overhead percentage of 32% which is about  $1/3k$ .

#### 4.4 Numerical Results and Discussion

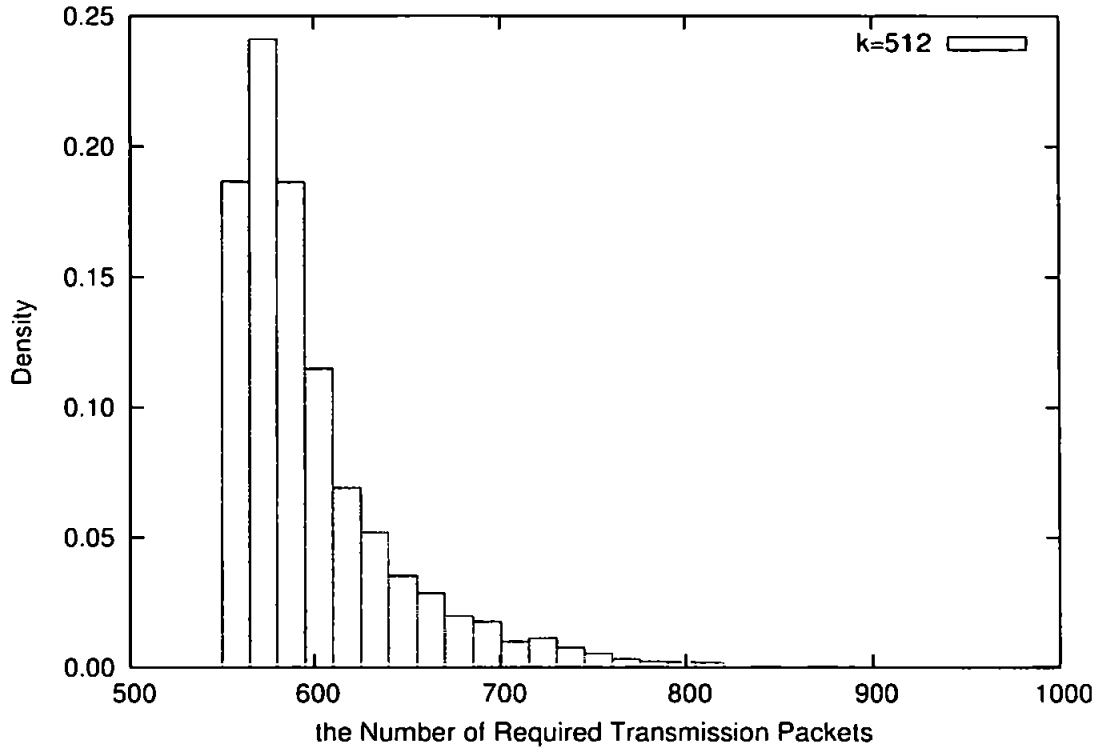


Figure 4.9: Histogram of the actual number of required transmission packets when the number of information packets is 512 and  $p = 0.5$

The examples on the relationship between the overhead percentage and the information file size have been concluded in Table 4.2 as follows:

$k$	10000	5000	1024	500	250	125
mean	10472	5320	1154	601	314	165
overhead %	4.72%	6.4%	12.7%	17.4%	25.6%	32%

Table 4.2: The Overhead Percentages on the different sizes of the information files, when  $p = 0.5$

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

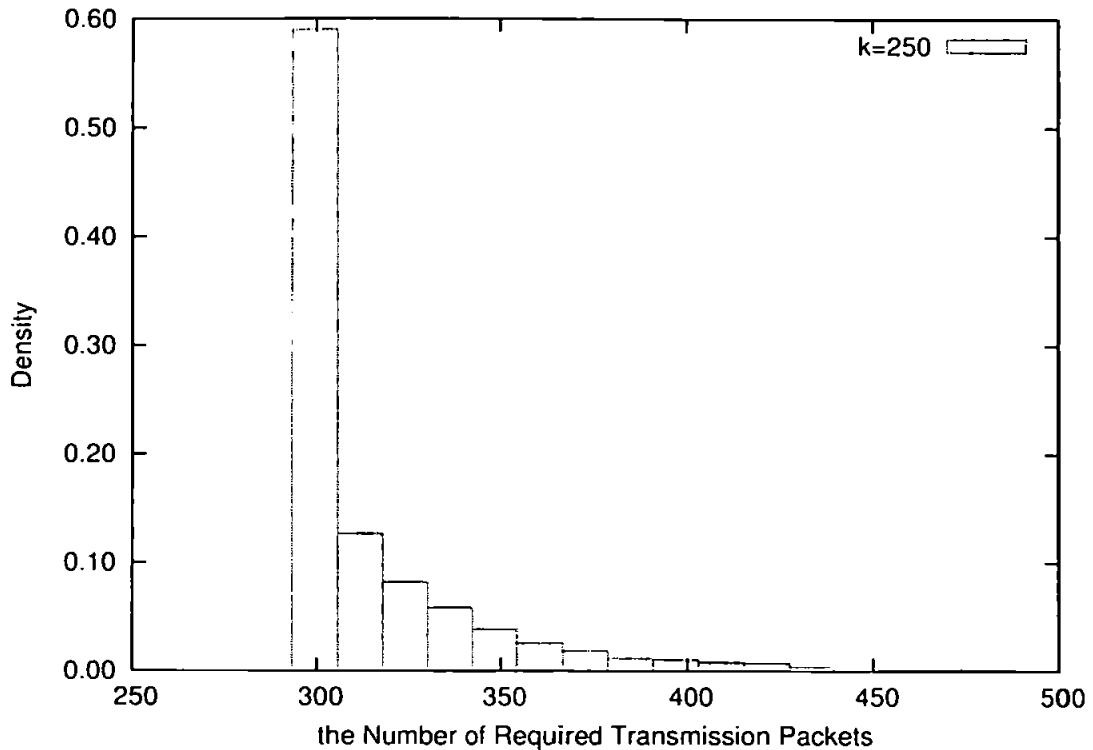


Figure 4.10: Histogram of the actual number of required transmission packets when the number of information packets is 250 and  $p = 0.5$

As stated above, short LT codes are not able to perform sub-optimally. Therefore, our second discussion is to answer the question:

*Whether a choice of a pre-code can make difference on the performance of short Raptor codes?*

To answer this question, we evaluate a short Raptor code of the length of 155 with two different pre-codes:

- a regular-LDPC code generated by the 3GPP standard [1].
- the binary-image RS code  $BRS(155, 125)$  with a very dense parity-check matrix

The simulation displayed in Figure 4.12 implies when the Raptor code is short, the pre-code does not significantly improve the performance dominated by its

#### 4.4 Numerical Results and Discussion

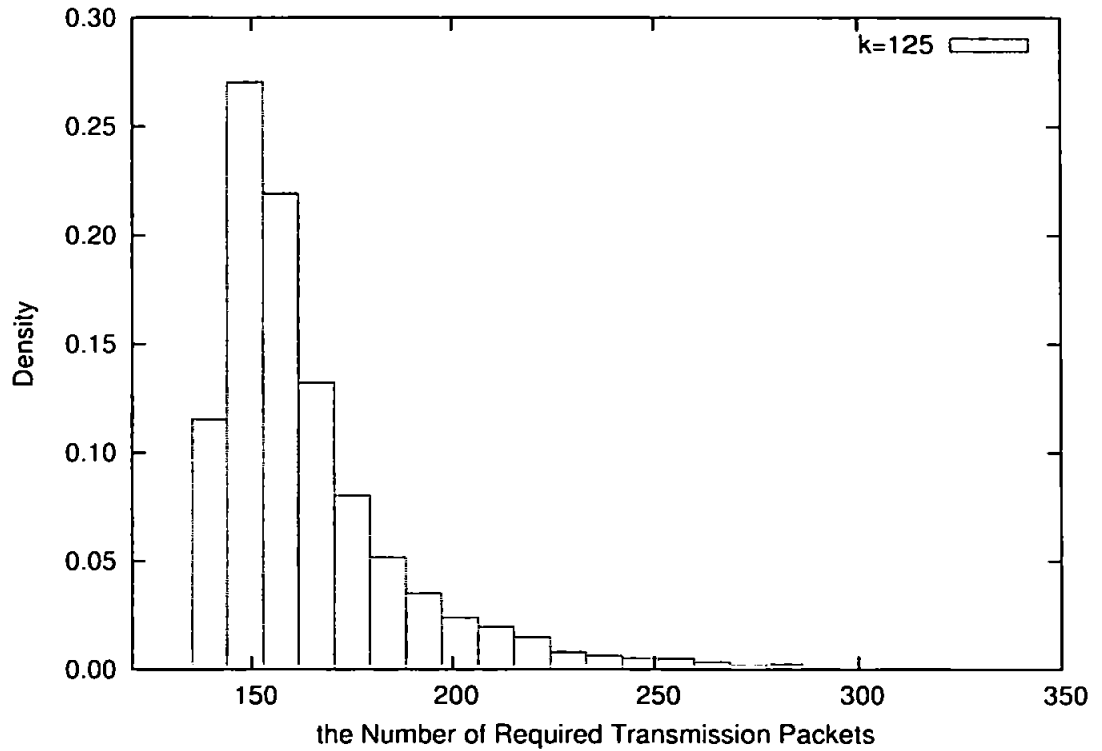


Figure 4.11: Histogram of the actual number of required transmission packets when the number of information packets is 125 and  $p = 0.5$

inner code (the LT code) part.

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

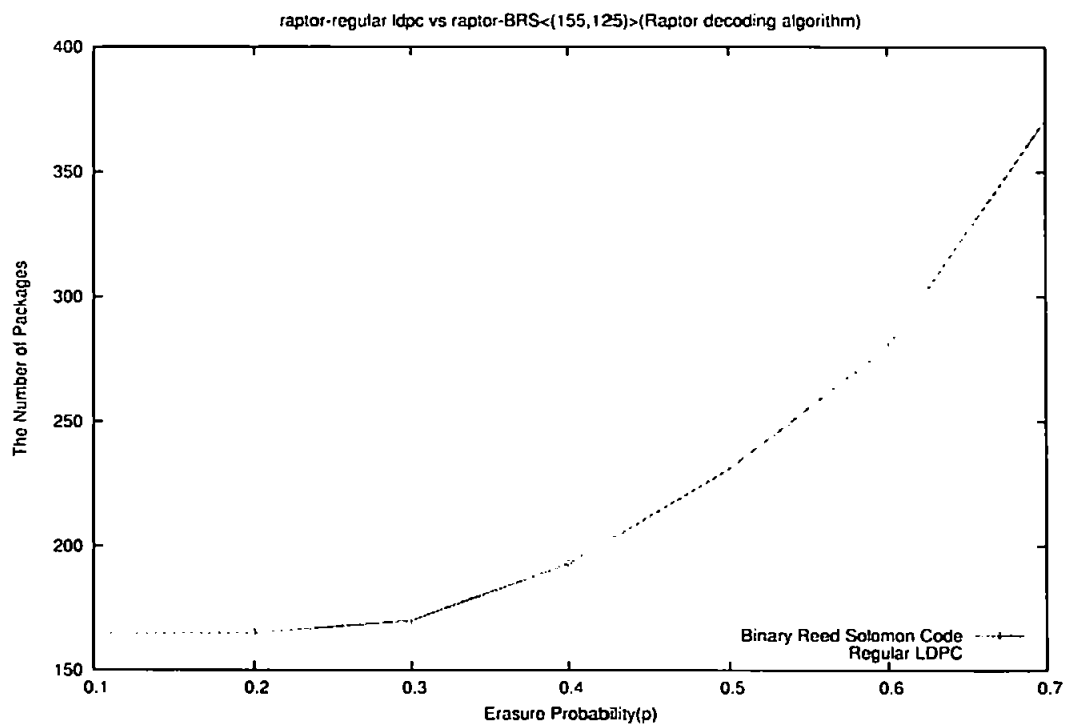


Figure 4.12: A Comparison on the Raptor Code Performances with Different Pre-encoding Scheme: Binary Reed-Solomon Codes vs Regular LDPC Codes



## 4.5 Summary

In this chapter, we study and investigate fountain codes including LT codes and Raptor codes. The matrix representations of LT codes and Raptor codes are introduced in this chapter. We have simulations on LT codes and Raptor codes. The findings are:

- When the value of  $k$  is more than 5000, the simulation results agree with the claim of  $k' \approx k$  asymptotically.
- When the number of the information packets is less than 5000, the overhead percentage is more than 10%. Especially when the information file is of a small size less than 200, the overhead percentage is up to 30% of the file size.
- Different pre-codes make nearly no difference on the performance of short Raptor codes.

In the presence of the difference overhead percentages caused by the different sizes of the information files, the reason behind it becomes of interest. The future work is suggested on a study of the random matrix theory.

#### 4. THE PACKET DATA TRANSMISSION SYSTEM OF FOUNTAIN CODES

---

*Big doors swing on little hinges.*

W. C. Stone

# 5

## Capacity Approaching Codes for the Binary Erasure Channel Using a Product Packetisation Method

### 5.1 Background

Network transmission is based on packet transmission. Multicast and broadcast are typical examples. The usual way to get around packet loss is to employ a protocol in which receiving parties acknowledge received packets. To limit the amount of feedback to the senders and the number of redundant packets sent to receivers, the system requires an erasure correcting code which is capable of providing a means to recover the lost/erased packets at receiver without the need for re-transmission.

One model for the network transmission is the binary erasure channel (BEC), which was introduced by Elias [19] in 1955. With a packet lost/erased due to

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

---

network congestion with probability  $p$ , the BEC has a channel capacity of  $1 - p$ . For a binary erasure channel, a rate  $R$  code encodes a message of size  $K$  into a transmission of  $K/R = N$  bits, so that the original message can be recovered from the unerased positions of the  $N$  bit message.

In a milestone paper, Luby *et. al.* [38] proposed the first realisation of a class of erasure codes – LT codes, which are termed rateless and are generated on the fly as needed. In [68], Shokrollahi introduced the idea of Raptor codes which adds an outer code to LT codes. Raptor codes have been established in order to solve the error floors exhibited by the LT codes. The study on LT codes and Raptor codes have been given in Chapter 4. It has been verified in Chapter 4 that fountain codes are sub-optimal erasure codes only when the information file is large enough with more than 10000 packets. Otherwise, the overhead percentage can be as high as 30% in order to recover an information file with its size of 125 packets, when the erasure probability  $p = 0.5$ .

On the other hand, low-density parity-check (LDPC) codes have been studied in [16, 39, 54] for application to the BEC. The iterative decoding algorithm, which is the same as Gallager's soft-decoding algorithm in [24], was implemented in [39]. Capacity-achieving degree distributions in design of LDPC codes for the BEC have been introduced in [39, 49, 67]. Finite-length analysis of LDPC codes over the BEC was accomplished in [16]. In that paper, the authors have proposed to use finite-length analysis to find good finite-length codes for the BEC. However, all these codes suffer from error-floor problems.

It is nature to consider Reed Solomon (RS) codes for the applications for network transmissions because they are optimal erasure codes. However, their non-binary and dense structures lead the enormously high-computational complexity in both the encoding process and the decoding process.

Bleichenbacher *et al.* [7] proposed an interleaved RS for the  $q$ -SC, where  $q = Q^m$ . Given a data sequence  $\{x_0, x_1, \dots, x_{k-1}\}$  in  $GF(q)$ , each  $x_i$  can be written as a vector  $(x_{i,1}, x_{i,2}, \dots, x_{i,m})$  of elements over  $GF(Q)$ . By applying an RS code  $(n, k)$  over  $GF(Q)$ , each vector  $(x_{0,j}, x_{1,j}, \dots, x_{k-1,j})$  can be encoded into a codeword  $(y_{0,j}, y_{1,j}, y_{n-1,j})$ . Then, we transmit the vector  $(y_0, \dots, y_{n-1})$  over the channel, of which each  $y_i$  contains  $(y_{i,1}, y_{i,2}, \dots, y_{i,m})$ . If an error occurs during the transmission at position  $i$ , it is highly possible that all the  $m$  components of

## 5.2 Product-packetisation Transmission Protocol

---

that position are corrupt. Therefore, when applying an algebraic decoder, the error locator polynomial should be the same. The radical idea of interleaved RS coding is the same as our proposed product packetisation structure in this chapter. The product packetisation is universal for all linear block codes. Also, we construct the RS code as an encoding-on-the-fly structure which is more effective and efficient for a data transmission.

The contribution of this chapter is to develop an coding scheme using incremental redundancy with an ML decoding algorithm. The algorithm has two key elements:

- the *Product Packetisation* structure
- the *In-place* decoding algorithm

The proposed algorithm has an overall computational complexity of  $O(K^{1.5})$ . Additionally, we propose the most powerful codes for the erasure channel, the Maximum Distance Separable (MDS) codes [43] together with a rateless transmission protocol.

The chapter is organised as follows. In Section 5.2, we describe the product packetisation method and the rateless transmission protocol. In Section 5.3 and 5.4, we give the implementation by using Reed Solomon (RS) block codes and analyse the computational complexities on both encoding and decoding. In Section 5.5, numerical results are given for these codes in comparison to LT codes, BCH codes and MDS codes. The conclusions are given in section 5.6.

## 5.2 Product-packetisation Transmission Protocol

### 5.2.1 Product Packetisation and De-packetisation

As we know, for network transmissions, a long message is usually broken up into a sequence of blocks which are then transmitted separately. In this chapter, we introduce a novel packetisation method – *Product Packetisation*. Instead of placing the symbols into the sequential packets, we first split the data into blocks

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

---

corresponding for the codewords of all erasure code, encode it and then packetise it by using symbols at the same positions of each codeword. Denoting the length of input as  $l_d$  and the payload size of each packet as  $l_p$ , the minimum number of packets needed can be calculated by  $l_d/l_p$ .

Hereafter, a RS block code  $C$  with this scheme is used with the number of information bits  $k = l_p$  and a nominal code rate  $R$ . The actual transmitted code rate is adaptive as described below. To ensure the efficiency of a transmission, the nominal code rate  $R$  is chosen to correspond to twice the average congestion probability

A network packet is composed of a payload and a fixed-length header. For a linear block code  $C(n, k)$ , there are  $k$  information packets and  $n - k$  parity-check packets, but the benefit of using the RS codes is that only  $k$  packets need to be received and these can be any  $k$  packets. For convenience, the first  $k$  symbols are termed information symbols. The reason for the variable number of parity-check packets is that the transmission uses *Incremental Redundancy (IR)*.

The  $i$ -th packet contains  $l_d$  information symbols whose positions are at  $j \cdot k + i$ , where  $j = 0, 1, \dots, (l_d - 1)$  and  $0 \leq i < k$ . The structure of the  $i$ -th packet is shown in Figure 5.1.

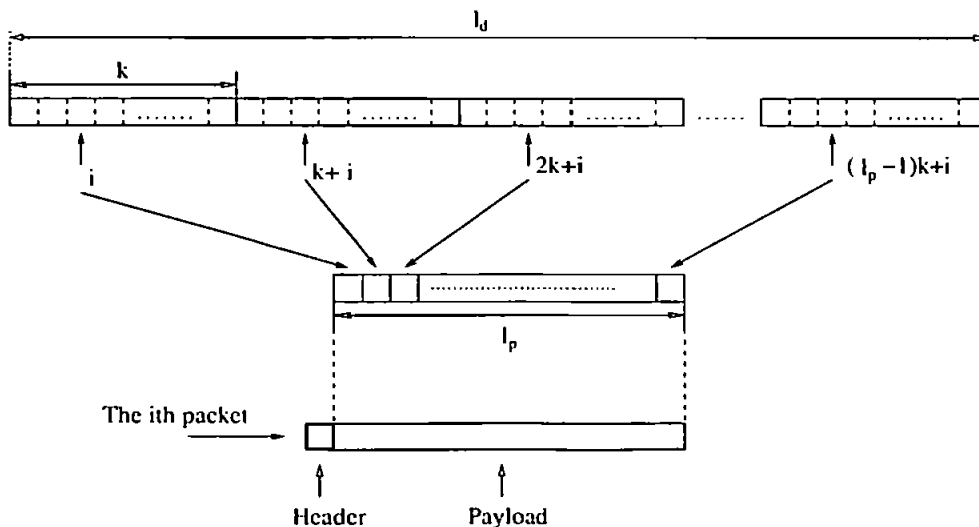


Figure 5.1: Packet construction for the  $i$ -th packet

## 5.2 Product-packetisation Transmission Protocol

The function of the header part is to handle the data block during the transmission. The header includes information which identifies the packet. To simplify the decoding, we use sequence numbers in the header to label each packet, i. e., if the “ $i$ ” and “ $i + s$ ” sequence number are received, it can be concluded that the packets  $j$ , where  $j = i + 1, \dots, i + s - 1$  have been erased during the transmission.

At the receiver, all the received information packets need to be restructured into a buffer  $X$  with length  $l_d$ . If the  $i$ -th packet is received, the symbols contained in the packet should be placed in the buffer at the positions of  $x_{j \times k + i}$ , where  $j = 0, \dots, k - 1$ ; if it has been erased during the transmission, ?’s will be placed in the positions of  $x_{j \times k + i}$ , where  $j = 0, \dots, k - 1$  to mark the symbols as erased. In this paper, we have assumed  $l_p \simeq n$ . This can be solved by Gaussian reduction and has a complexity of the order  $O(n^3) = O(l_d^{1.5}) = O(K^{1.5})$ .

### 5.2.2 Protocol Description

The packets are transmitted continuously and at each destination host, they are depacketised and decoded into a buffer with a length of  $l_d$ . If after  $k + \theta$  packets have been sent, all the information packets are received, a positive acknowledgement (ACK) is sent and the transmission of the current codeword ends. The protocol structure is shown in Figure 5.2.

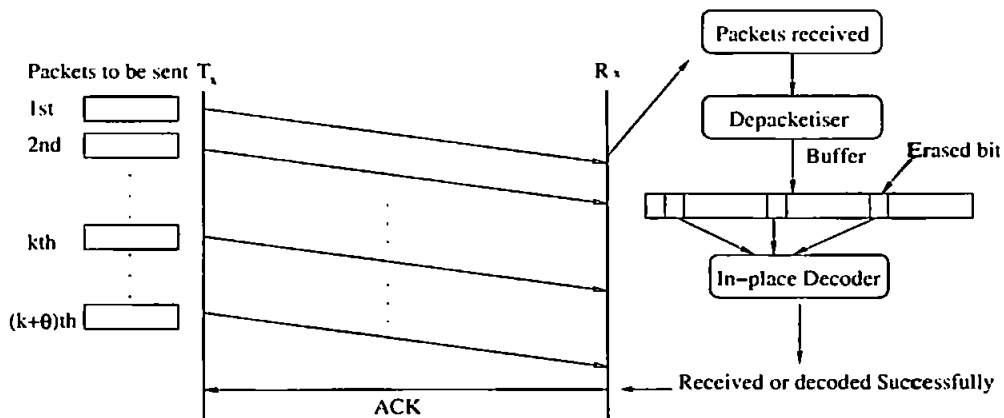


Figure 5.2: The Protocol Structure

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

---

Consider an  $(n, k)$  RS code  $C$ , which has  $n$  symbols and  $k$  information symbols. After  $k$  symbols have been transmitted, transmission of the  $n - k$  symbols, continues one symbol at a time until all the acknowledgements from destination hosts have been received. If after transmitting  $n$  symbols, not all the acknowledgements have been received, transmission is repeated, one symbol at a time, starting with the first symbol. The total number of symbols transmitted  $S_{\text{total}} = z + n\rho$ , where  $\rho$  is the number of repeated blocks and  $z$  is the number of symbols which is transmitted in the final block.

### 5.3 Rateless Reed Solomon Codes

Reed Solomon (RS) code, as known as the most popular *maximum distance separable* (MDS) code, has been considered as one of the optimal erasure codes. The RS code is designated as  $\mathcal{C}_{RS}$  with its length  $n = q^m - 1$ , coordinates  $k$  and  $d_{\min} = n - k + 1$  over the Galois Field  $GF(q^m)$  and written as  $\mathcal{C}_{RS}(n, k, d_{\min})$ .

**5.1 Definition (MDS Codes).** [43] For a linear code  $\mathcal{C}(n, k, d_{\min})$  over any field, maximum distance separable (MDS) codes have the maximum possible minimum distance satisfying the equation of

$$d_{\min} = n - k + 1. \quad (5.1)$$

**5.1 Lemma.** A code  $\mathcal{C}$  is a MDS code if and only if every set of  $n - k$  columns of its parity check matrix  $H$  are linearly independent.

*Proof.* Following the definition of the minimum Hamming distance  $d_{\min}$  of a linear block code  $\mathcal{C}$ , the value of  $d_{\min}$  equals to the smallest positive number of columns of its  $H$  which are linearly dependent. Whereas, if and only if no  $n - k$  or fewer columns of  $H$  are linearly dependent, the code  $\mathcal{C}$  then has its minimum distance of  $d_{\min} = n - k + 1$  and therefore it is a MDS code.  $\square$

In this section, we describe a way to encode the input symbols, using the “encoding-on-the-fly”. This is one of the main strength of LT codes. The disadvantage of LT codes is that they are not very powerful codes.



Linear block codes can be generated from their parity-check matrix, denoted by  $\mathbf{H}$ , which is usually a row-echelon reduced  $\mathbf{H}$  matrix. Any matrix of rank  $k$  can be described by the parity-check equations  $\mathbf{h}_i$ , where  $i = 0, 1, \dots, n - k - 1$ .

Each parity-check equation  $\mathbf{h}_i$  can generate a parity-check  $p_i$  independently. The variable rate can be realised by generating parity-checks only if required.

For an  $(n, k)$  RS code  $\mathbf{C}$  over  $GF(q)$ , and  $n = q - 1$  and  $q$  is a prime number or a power of a prime number and  $\alpha$  is a primitive element of  $GF(q)$ . An RS code has a minimum distance  $d_{\min} = n - k + 1$  and can correct any erasure pattern involving up to  $n - k$  erasures. The parity check matrix of an RS code can be represented by:

$$\mathbf{H} = \begin{pmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{d_{\min}-1} & \alpha^{2(d_{\min}-1)} & \dots & \alpha^{(n-1)(d_{\min}-1)} \end{pmatrix} \quad (5.2)$$

This  $\mathbf{H}$  matrix is not efficient for our propose because every parity check symbol requires a calculation of using  $n$  symbols. From Theorem 11 – 9 in [43], there exist cyclic MDS codes over  $GF(q)$ . In order to efficiently perform encoding on-the-fly, we will use the cyclic form of the  $\mathbf{H}$  matrix. For this, the parity check polynomial  $h(x)$  of the RS code is used.

Define the set of powers of roots as  $\Gamma = \{1, 2, \dots, d_{\min} - 1\}$

$$h(x) = \prod_{i=0, i \notin \Gamma}^{n-1} (x - \alpha^i) = \sum_{i=0}^k \beta_i x^i. \quad (5.3)$$

The  $\mathbf{H}$  matrix can be written as:

$$\mathbf{H} = \begin{pmatrix} \beta_0 & \beta_1 & \dots & \beta_k & 0 & \dots & \dots \\ 0 & \beta_0 & \beta_1 & \dots & \beta_k & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \beta_0 & \beta_1 & \dots & \beta_k \end{pmatrix} \quad (5.4)$$

Each of the  $\beta_i$  is an element from  $GF(q)$ . For binary transmission,  $q = 2^m$  can be used to construct the RS code. As an example, for  $q = 2^4 = 16$ ,  $\alpha$  has

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

---

a binary vector representation of  $[0, 1, 0, 0]$  and multiplication by  $\alpha$  corresponds to multiplication by the binary matrix  $\begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \end{pmatrix}$ . And also  $\beta$  can be obtained by  $\alpha^i$ . Therefore, by replacing the  $GF(16)$  values in (5.4), a binary form of the  $H$  matrix can be obtained. Encoding on-the-fly can be efficiently performed in this case by using parity check equations involving a maximum of  $k + 1$  bits. On average there will be  $\frac{k+1}{2}$  bits;  $\frac{m}{2}(k + 1)$  per  $GF(q)$  symbol.

Assume the average weight of the parity-check equations is  $\omega$ . For a single parity-check generation, the encoding complexity is  $O(\omega)$ . If the erasure probability of the channel is  $\delta$ , the number of erasures, denoted as  $\varepsilon$ , is  $\varepsilon = \frac{k\delta}{1-\delta}$  after receiving  $k$  unerased symbols. The average number of parity symbols calculated equals  $\varepsilon$ . Therefore, the average overall encoding complexity can be written as:  $\varepsilon \cdot O\left(\frac{m}{2}(k + 1)\right)$ . For binary  $H$  matrix, the encoding complexity is  $O\left(\left(\frac{k\delta}{1-\delta}\left(\frac{k+1}{2}\right)\right)m\right)$ .

### 5.4 ML Decoding Algorithm on Rateless RS Codes

Let  $\mathbf{x}'$  denote the received vector. According to [16], optimal decoding is equivalent to solving the linear system, shown in (5.5). In our case, on average  $k + \varepsilon$  packets are transmitted before  $k$  unerased packets by each destination host.

Accordingly the following set of equations need to be solved from the  $H$  matrix shown by (5.4).

$$\sum_{i=0}^{k+\varepsilon-1} h_{ij}x'_i = 0, \quad j = 0, \dots, c - 1. \quad (5.5)$$

For example, for an RS code, we have (5.6).

$$\begin{pmatrix} \beta_0 & \beta_1 & \dots & \dots \\ 0 & \beta_0 & \beta_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \beta_0 & \dots \end{pmatrix} \begin{pmatrix} x'_0 \\ x'_1 \\ \vdots \\ x'_{k+\varepsilon-1} \end{pmatrix} = \mathbf{0}. \quad (5.6)$$

## 5.4 ML Decoding Algorithm on Rateless RS Codes

for  $\epsilon$  erasures  $0 \leq \epsilon \leq n - k - 1$ .

This linear system can be used to solve for at most  $n - k$  erasures in the case of RS codes. If the equation (5.6) has a unique solution, an optimal algorithm is possible. The Gaussian Elimination (GE) algorithm is considered as an optimal algorithm over the BEC, but has a complexity of  $O(N^3)$ .

We propose a reduced complexity ML algorithm – In-place Algorithm [10, 74] (described in Chapter 2.4) by eliminating the column-permutations required combined with a two dimensional code so as to achieve a reduction in decoder complexity. The conventional linear system equation of  $\mathbf{H}\mathbf{x}^T = \mathbf{0}$  can be written for all  $i \in \{0, \dots, N - K - 1\}$  as:

$$\sum_{j=0}^{N-1} h_{i,j}x'_j = \sum_{j \in \mathcal{E}} h_{i,j}x'_j + \sum_{j \in \bar{\mathcal{E}}} h_{i,j}x'_j = 0 \quad (5.7)$$

The erased positions  $\mathcal{E}$  can be solved as a function of the non-erased positions  $\bar{\mathcal{E}}$ :

$$\sum_{j \in \mathcal{E}} h_{ij}x'_j = \sum_{j \in \bar{\mathcal{E}}} h_{ij}x'_j \quad (5.8)$$

This is solved by the GE method and has a complexity ranging from  $O(N^3)$  to  $O(N^2)$ , depending on the algorithm used. The proposed two dimensional structure uses a code of length  $N = n^2$  and the coded bits are transmitted as  $n$  packets of length  $n$  bits. Each packet  $s$  contains all bit positions  $s + pn$  in the transmitted codeword, with  $0 \leq p < n$ . This establishes a regular, two-dimensional structure on  $\mathcal{E}$ : if bit position  $s$  is erased by a packet loss, then so are all the bit positions  $s + pn$ . In this case, if we define  $\mathcal{E}' = \{s \in \mathcal{E} | 0 \leq s < n\}$ , Equation (5.8) becomes

$$\sum_{p=0}^{n-1} \sum_{s \in \mathcal{E}'} h_{i,s+pn}x'_{s+pn} = \sum_{p=0}^{n-1} \sum_{s \in \bar{\mathcal{E}}'} h_{i,s+pn}x'_{s+pn} \quad (5.9)$$

By swapping the order of summation and writing the index  $i = l + \nu m$ , where

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

---

$mn = N - K$ ,  $l \in \{0, \dots, m - 1\}$  and  $\nu \in \{0, \dots, n - 1\}$ , we obtain:

$$\sum_{j \in \mathcal{E}'} \sum_{p=0}^{n-1} h_{l+\nu m, s+pn} x'_{s+pn} = \sum_{j \in \bar{\mathcal{E}}'} \sum_{p=0}^{n-1} h_{l+\nu m, s+pn} x'_{s+pn} \quad (5.10)$$

We use a code such that  $h_{l+\nu m, s+pn} = \delta(\nu, p) h_{l, s}$  with  $\delta(\nu, p) = 1$  if  $\nu = p$  and zero otherwise. Equation (5.10) can be written in this case as:

$$\sum_{j \in \mathcal{E}'} h_{l, s} \sum_{p=0}^{n-1} \delta(\nu, p) x'_{s+pn} = \sum_{j \in \bar{\mathcal{E}}'} h_{l, s} \sum_{p=0}^{n-1} \delta(\nu, p) x'_{s+pn} \quad (5.11)$$

$$\sum_{s \in \mathcal{E}'} h_{l, s} x'_{s+\nu n} = \sum_{s \in \bar{\mathcal{E}}'} h_{l, s} x'_{s+\nu n} \quad (5.12)$$

$\forall l \in \{0, \dots, m - 1\}$  and  $\nu, s \in \{0, \dots, n - 1\}$ . This can be solved by the In-place algorithm and has a complexity of the order  $O(n^3) = O(N^{1.5}) = O(K^{1.5})$ .

An *ACK* is sent when all ACKs have been received, a total of  $n\rho + z$  symbols will have been transmitted, whose  $\rho$  is an integer. On average  $n\rho + z = k + c$ . In this sense this is a rateless erasure correcting system.

### 5.5 Numerical Results and Discussion

We have evaluated the performance of LT codes and cyclic codes for  $N = 225$  and  $N = 3969$ . For  $N = 225$ , the LT code with its recovery decoder produces very poor performance. To improve the performance of the LT code we have replaced the recovery decoder with the ML decoder. The LT code (225, 105) was divided into 15 packets with a packet-size of 15 bits and the cyclic code (225, 105) was arranged as a  $15 \times 15$ , two dimensional code with the same packet-size. The cyclic code (225, 105) has a generator polynomial

$$g(x) = 1 + x^{15} + x^{30} + x^{60} + x^{120}, \quad (5.13)$$

## 5.5 Numerical Results and Discussion

which is a replication of the BCH (15, 7, 5) code with  $g(x) = 1 + x + x^2 + x^4 + x^8$ . The cyclic code is capable of correct decoding up to and including 4 erased packets. As  $1 + x^{15}$  divides  $1 + x^{225}$ , the decoder for this cyclic code can be implemented with the reduced complexity decoder. As shown in Figure 5.3, the performance of the cyclic code (225, 105) is significantly better than that of the LT (225, 105) code. When  $p = 0.1$ , the result of the cyclic code (225, 105) is over 2 orders of magnitude better than that of the LT (225, 105).

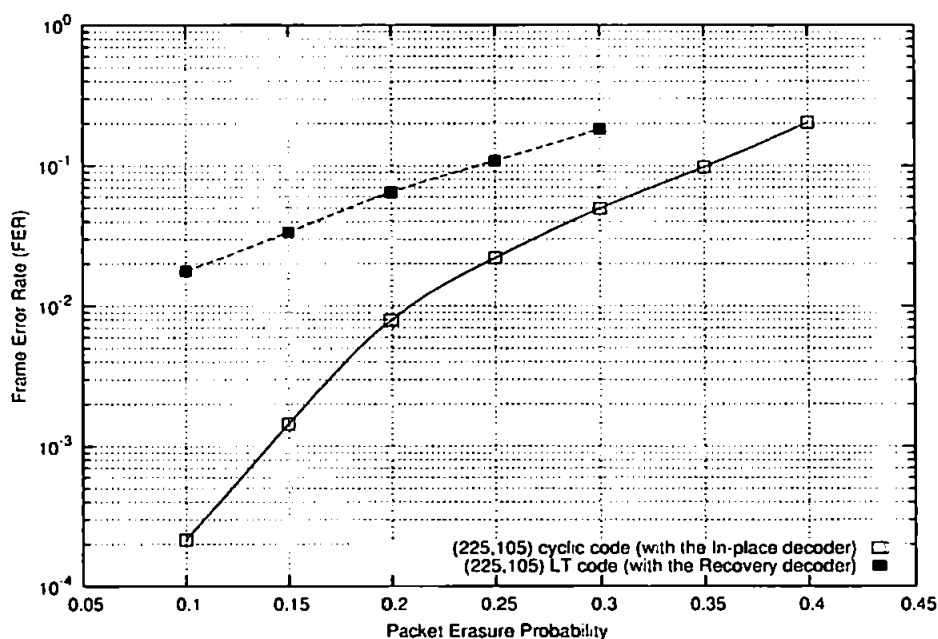


Figure 5.3: FER vs the erasure probability of erased packets for the cyclic code (225, 105) and the LT (225, 105) code over the Erasure Channel, both using ML decoding

To verify the decoding complexity of  $O(N^{1.5})$ , we implemented a longer code, the cyclic code (3969, 2016) and the LT (3969, 2016) code. The cyclic code (3969, 2016) is based upon the generator polynomial of the BCH (63, 32, 12) code with an expansion factor of 63. The generator polynomial of the BCH (63, 32, 12) code is  $g(x) = 1 + x + x^7 + x^9 + x^{12} + x^{14} + x^{15} + x^{16} + x^{18} + x^{19} + x^{20} + x^{21} + x^{26} + x^{28} + x^{30} + x^{31}$ .

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

---

Consequently, the cyclic code (3969, 2016) has a generator polynomial of:

$$g(x) = 1 + x^{63} + x^{441} + x^{567} + x^{756} + x^{882} + x^{945} + x^{1008} + x^{1134} + x^{1197} \\ + x^{1260} + x^{1323} + x^{1638} + x^{1764} + x^{1890} + x^{1953}$$

The RS (63, 32, 32) is generated by the primitive polynomial  $p(x) = 1 + x + x^6$  in  $GF(2^6)$ .

The simulated performance is shown in Figure 5.4. On a standard 1GHz PC, the LT (3969, 2016) code took 775ms to encode each codeword and 141ms to decode each codeword for  $p = 0.4$  (using the Recovery Decoder). In contrast, the cyclic code (3969, 2016) took only 5ms to encode each codeword and 8ms to decode each codeword using the In-place decoder for  $p = 0.4$  (the decoding time for this decoder is independent of the value of  $p$ ). Noting that RS codes are MDS codes whose minimum distance  $D = N - K + 1$ , Figure 5.4 shows the simulation performance of the RS-assembled cyclic code is virtually identical to the theoretical bound for RS (MDS) codes.

Interestingly, the performance is more than one order of magnitude better than that of the BCH-assembled cyclic code, demonstrating the loss attributable to the BCH code being non MDS. Figures 5.5 and 5.6 show the histograms of the number of transmitted packets required when  $p = 0.25$ . (Assuming an acknowledgement is sent to prevent further packets being transmitted once correct decoding is achieved) The average number of transmitted packets,  $n_a$ , needed when using the Reed Solomon cyclic code is 2688 and that for BCH cyclic code is 2814. This demonstrates that the Reed Solomon code achieves the erasure channel capacity of  $k = n_a(1 - p)$  since  $k = 2016$  and  $(1 - p) = 0.75$ .

Figure 5.7 shows a performance of a long code, the cyclic(76500,39300). The simulation performance of the RS-assembled cyclic code is around half order of magnitude better than that of the BCH-assembled cyclic code. From the histograms of Figures 5.8 and 5.9, when  $p = 0.42$ , for using the BCH cyclic code and the RS cyclic code, the average numbers of transmitted packets are 68385 and 67758 respectively.

## 5.5 Numerical Results and Discussion

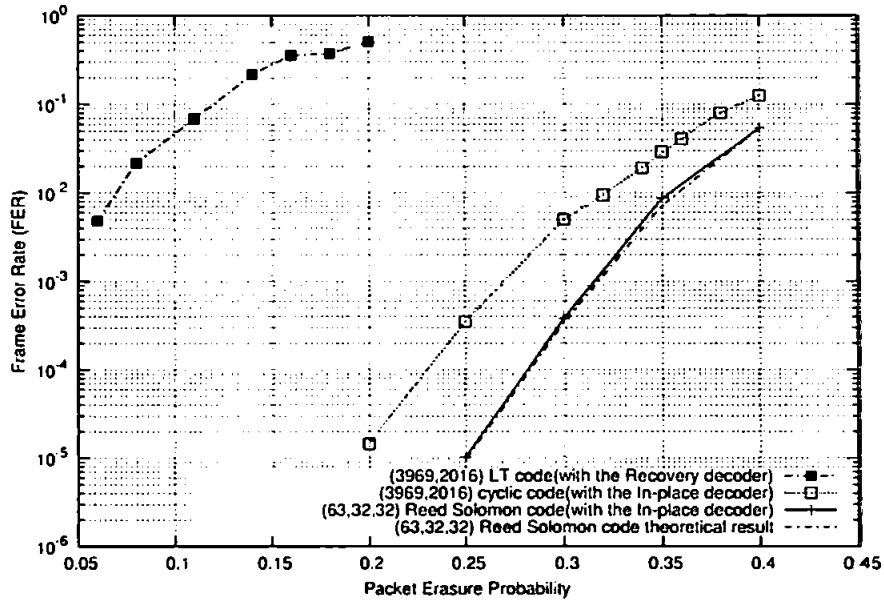


Figure 5.4: FER vs the erasure probability of erased packets for the cyclic code (3969,2016) based on the BCH(63,32,12) and the Reed Solomon(63,32,32), the LT(3969,2016) code and theoretical result over the Erasure Channel

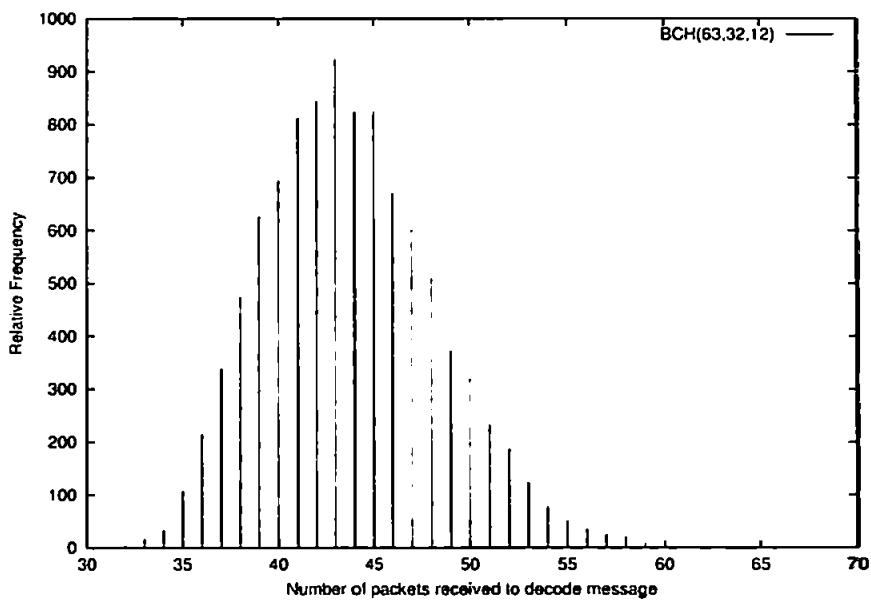


Figure 5.5: Histogram of the cyclic code (3969,2016) based on the BCH(63,32,12) when  $p = 0.25$ , mean = 44.7

## 5. CAPACITY APPROACHING CODES FOR THE BINARY ERASURE CHANNEL USING A PRODUCT PACKETISATION METHOD

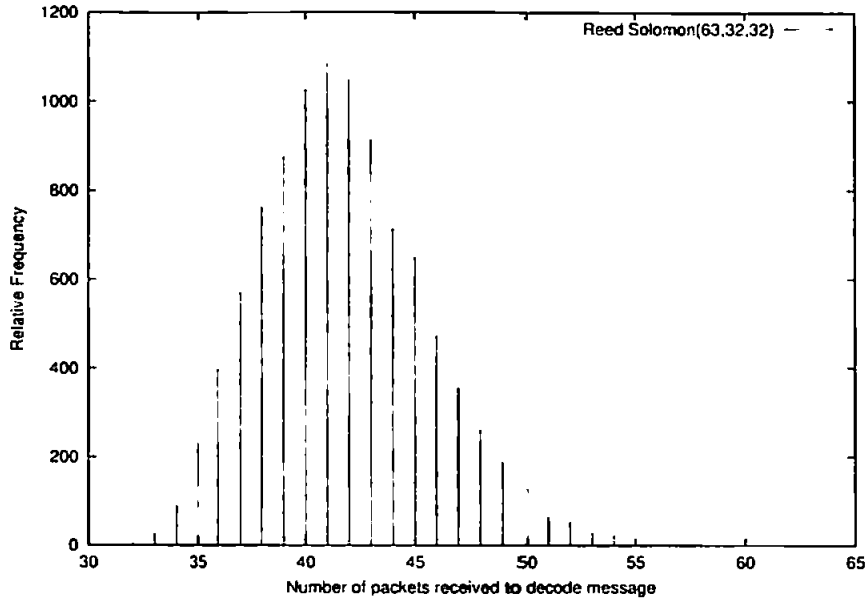


Figure 5.6: Histogram of the cyclic code(3969, 2016) based on the RS(63, 32, 32) when  $p = 0.25$ , mean = 42.7

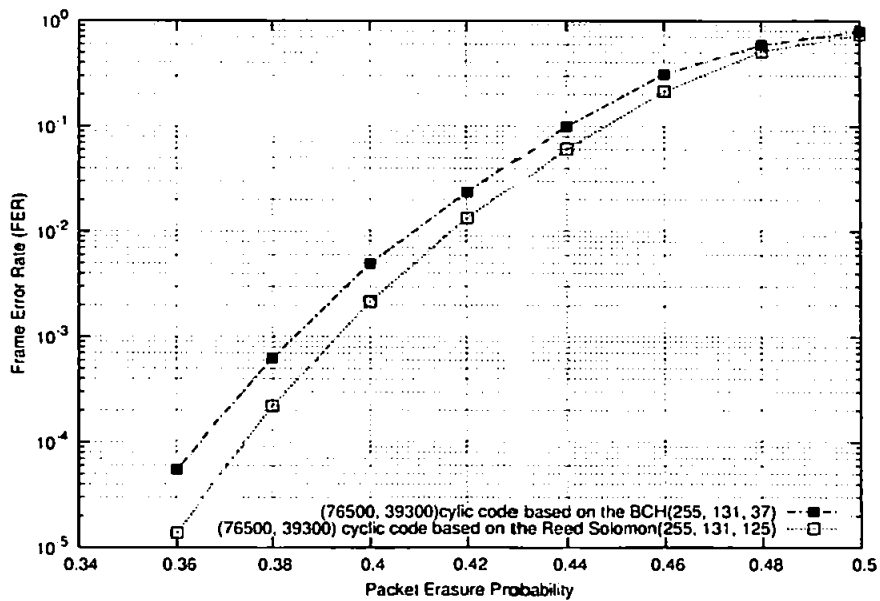


Figure 5.7: FER vs the erasure probability of erased packets for the cyclic code (76500, 39300) based on the BCH (255, 131, 37) and the RS (255, 131, 125) over the Erasure Channel



## 5.5 Numerical Results and Discussion

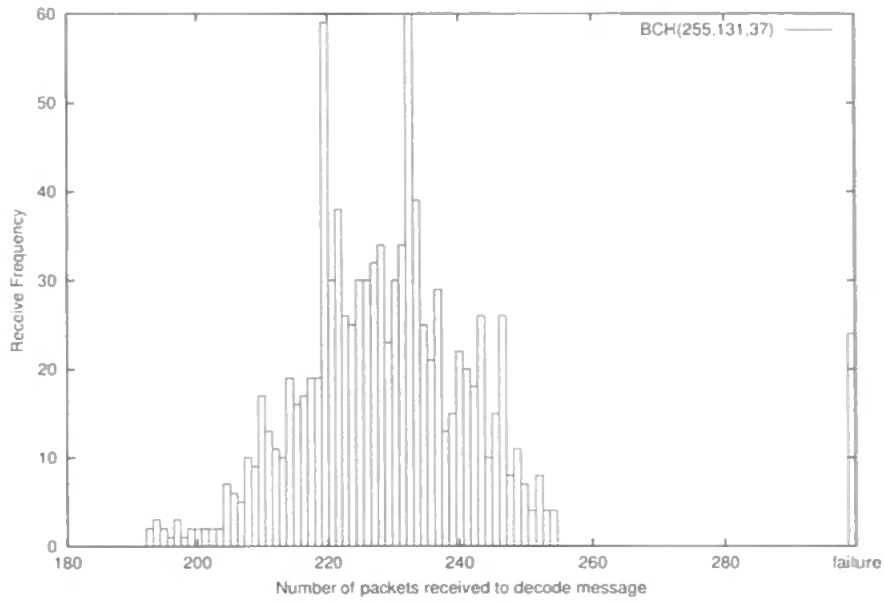


Figure 5.8: Histogram of the cyclic code(76500,39300) based on the BCH(255, 131, 37) when  $p = 0.42$ , mean = 227.95

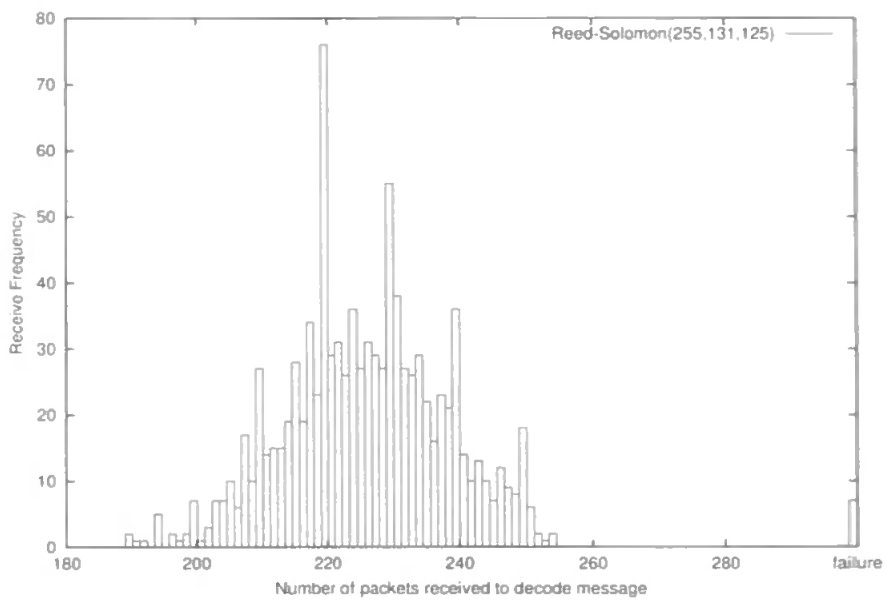


Figure 5.9: Histogram of the cyclic code(76500,39300) based on the Reed-Solomon(255,131,125) when  $p = 0.42$ , mean = 225.86

## 5.6 Summary

In this chapter, we present a new coding and decoding algorithm for packet networks modelled as a BEC. The algorithm combines an ML decoder with a two dimensional code and enables the use of optimum MDS codes (RS codes). In the past, iterative decoding algorithms, although non-optimum, have been used as a good tradeoff between performance and computational complexity. With the new approach, decoding is ML and the computational complexity is reduced to  $O(K^{1.5})$ . This is less complexity than the case for iterative decoders. In addition, compared with LT codes, the new decoding algorithm with algebraic codes achieves a significant improvement in performance, and approaches theoretical limits when using RS codes. A method of realising a rate-adaptive transmission method has been described together with analysis and results which demonstrate that RS codes achieve the capacity limits of the erasure channel.

*Simple things should be simple, complex things should be possible.*

Alan Kay

# 6

## Concatenated Reed-Solomon Coding with Hard-decision for the Rayleigh Fading Channel

### 6.1 Background

The Rayleigh fading channel is widely used as a model of wireless communications. Channel coding techniques are a powerful tool to improve the reliability and efficiency of wireless communications. From [25], [33] and [27], the performance on Rayleigh fading channels and Turbo codes designed for the particular channel have been explored and given. In [31], irregular LDPC codes have also been applied to an uncorrelated flat Rayleigh fading channel, and shown to outperform Turbo codes over a wide range of mobile speeds.

Due to the existence of error bursts in the Rayleigh fading channel, the erasure correcting codes have been considered as the code candidates for the application

## 6. CONCATENATED REED-SOLOMON CODING WITH HARD-DECISION FOR THE RAYLEIGH FADING CHANNEL

---

of this specified channel. In this chapter, we have proposed the classic erasure correcting code – the Reed Solomon (RS) code in reason of the random bursts. The RS code constitutes an efficient class of linear codes using multi-bit symbols and has the strong capability of correcting/detecting symbol errors and correcting symbol erasures.

It is well-known that an RS code  $\mathcal{C}(n, k, d)$ , where  $n$  is the code length,  $k$  is the information length, and  $d$  is the Hamming distance of it, is capable of correcting up to  $t = \lfloor \frac{n-k}{2} \rfloor$  random symbol errors, and correct up to  $n - k$  symbol erasures. The classical algorithms of Berlekamp [4] and Massey [43] can correct  $t$  errors and  $\epsilon$  erasures when  $2t + \epsilon \leq n - k$ , which can achieve the error bound  $Pr = \binom{n-k+1}{2}$  with running time  $O(n^2)$ . In [72], it was presented that a polynomial time list decoding algorithm for RS codes can correct more than  $\frac{n-k}{2}$  errors, provided  $k < n/3$ . Using the Roth & Ruckenstein [65] algorithm, the same bound  $Pr$  can be achieved with running time  $O(n^2 \log^2 n)$ . Since more erasures than errors can be corrected, it is advantageous to determine the reliability of the received RS-coded symbols and to erase the low-reliability symbols prior to the decoding process.

This chapter is organised as follows. Section 6.2 briefly reviews the Rayleigh fading channel. In Section 6.3, we describe the system arrangement. In Section 6.4, we give an analysis of a concatenated RS code over the Rayleigh fading channel. In Section 6.5, numerical Frame Error Rate (FER) results are given for these codes in comparison to soft-decision decoding of LDPC codes as a function of the overall concatenated code rate. The summary of this chapter is given in Section 6.6.

## 6.2 System and Channel Model

### 6.2.1 Rayleigh Distribution

The Rayleigh fading channel is associated with the Rayleigh distribution which is a continuous probability distribution. This distribution is named after *Lord Rayleigh*, which is built up based on two normally distributed variables, both of which are with an equal variance [50].

---

## 6.2 System and Channel Model

Assume  $u_1$  and  $u_2$  as two *Gaussian* variables and both of them are distributed according to  $\mathcal{N}(0, \sigma^2)$ <sup>\*</sup>, and hereafter the Rayleigh random variable is written as:

$$u = \sqrt{u_1^2 + u_2^2} \quad (6.1)$$

The probability density function (PDF) can be derived from the PDF of the Gaussian variables which has been given in many textbooks [55].

$$PDF(u) = \begin{cases} \frac{u}{\sigma^2} e^{-\frac{u^2}{2\sigma^2}} & u > 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.2)$$

Its corresponding cumulative density function (CDF) can be derived by integrating the PDF.

$$CDF(u) = \begin{cases} 1 - e^{-\frac{u^2}{2\sigma^2}} & u > 0 \\ 0 & \text{otherwise} \end{cases} \quad (6.3)$$

The mean value is  $E(u) = \sigma\sqrt{\frac{\pi}{2}}$  and the variance is  $var(u) = (2 - \frac{\pi}{2})\sigma^2$ .

Figure 6.1 and Figure 6.2 illustrate the PDF graph and the CDF graph of the Rayleigh random variables for different values of  $\sigma$ , respectively.

In this chapter, we only consider the Rayleigh random variable generated from two individual zero-mean Gaussian random variables rather than the generalised Rayleigh random variable.

---

<sup>\*</sup> $\mathcal{N}(m, \sigma^2)$  denotes the PDF of Gaussian random variables with  $m$  as the mean value and  $\sigma$  is the variance.

## 6. CONCATENATED REED-SOLOMON CODING WITH HARD-DECISION FOR THE RAYLEIGH FADING CHANNEL

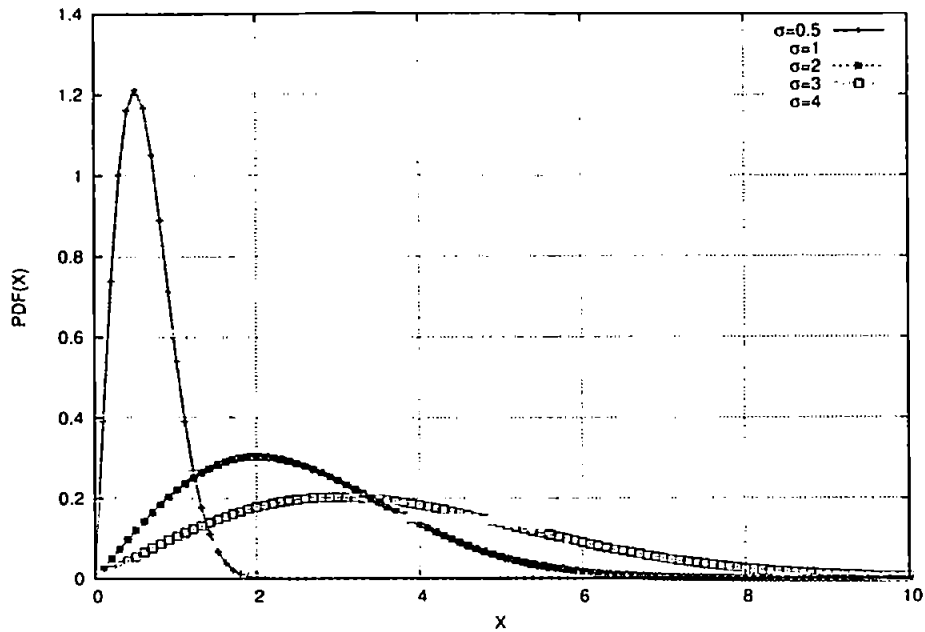


Figure 6.1: The PDF of the Rayleigh random variables for different values of  $\sigma$

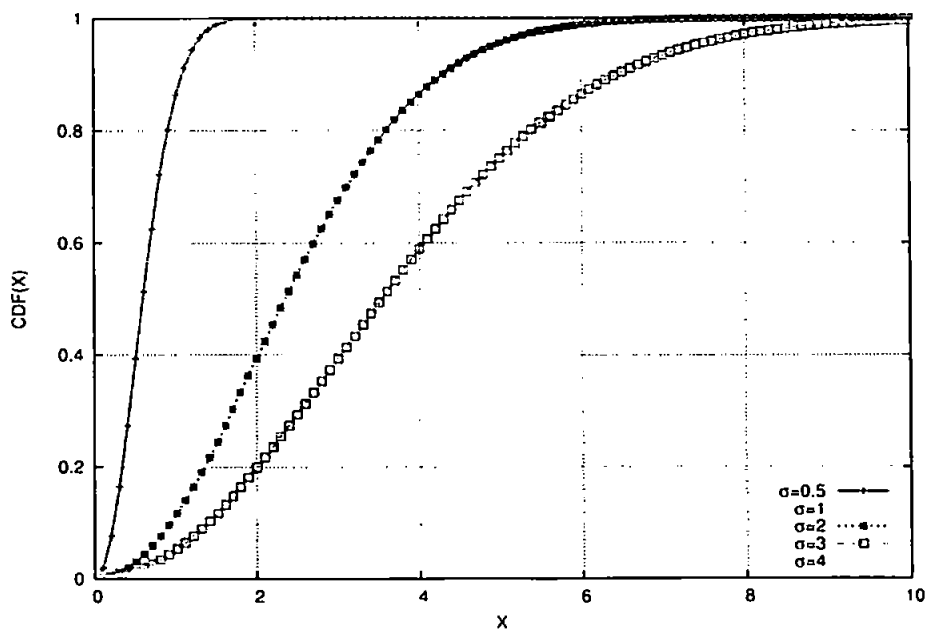


Figure 6.2: The CDF of the Rayleigh random variables for different values of  $\sigma$

### 6.2.2 Rayleigh Fading Channel

In this chapter, we assume the Rayleigh fading channel is a frequency-non selective, slowly fading channel, which means the fading condition of the channel is processed as a constant during at least one signalling interval [55].

Denote the transmitted sequence  $\mathbf{x} = \{x_0, x_1, x_2, \dots, x_i\}$  passing through a discrete time Rayleigh fading channel with an additive white Gaussian noise, where  $i$  using as a bit index as usual.  $x_i$  is defined as a BPSK-modulated bit with its amplitude as  $\pm \sqrt{Amp(x_i)}$ . At receiver, the received discrete-time based band-pass signal, which is designated as  $\mathbf{y} = \{y_0, y_1, y_2, \dots, y_i\}$ , can be derived from  $\mathbf{x}$  by the relationship:

$$y_i = \alpha_i x_i + n_i \quad (6.4)$$

where  $\alpha_i$  is called the Rayleigh distributed fading coefficient with  $Amp(\alpha_i^2) = 1$  and  $n_i$  is a complex white noise sample with its variance  $N_0/2$  per dimension.

The probability density function (pdf) of the output  $y$  can be described as:

$$p(y|\omega, \alpha) = \frac{1}{\sqrt{2\pi}\delta^2} \exp\left(-\frac{(y - \omega \cdot \alpha)^2}{2\delta^2}\right). \quad (6.5)$$

where  $\omega = (1 - 2x)$  is the binary input after the BPSK modulation,  $\delta^2 = (\frac{1}{2R} \cdot (E_b/N_0))$ , and  $R$  is the code rate.

## 6.3 the System Arrangement

The radical idea for the proposed system is to recover the severe faded transmitted symbols by  $C_2[n_2, k_2, d_2]$  and correct the errors by  $C_1[n_1, k_1, d_1]$ .

Figure 6.3 gives the arrangement of the system.

We deploy the product packetisation method to arrange symbols into packets. Instead of placing the coded symbols into sequential packets, the data is firstly split into blocks corresponding for the codewords of all erasure code. Secondly, we encode the data block by block and then packetised them by using symbols at the positions of each codeword. More details has been clearly introduced in Chapter 5.

## 6. CONCATENATED REED-SOLOMON CODING WITH HARD-DECISION FOR THE RAYLEIGH FADING CHANNEL

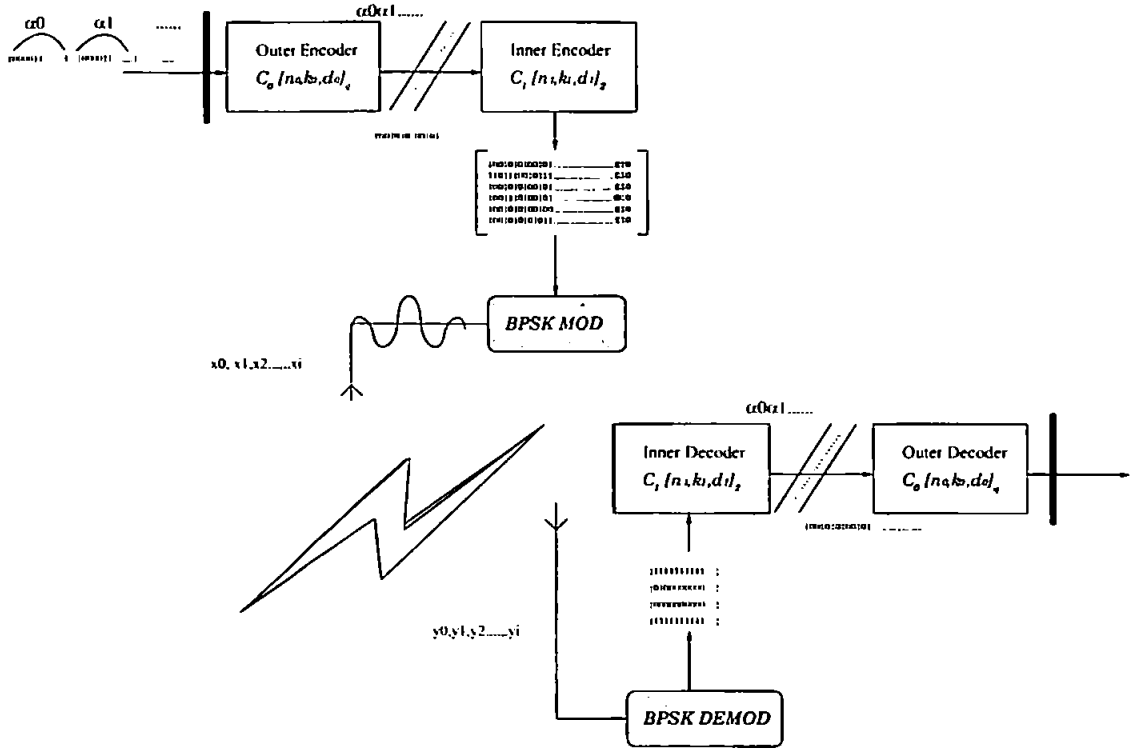


Figure 6.3: an Overview of the Proposed Transmission System

Instead of placing the symbols into the sequential packets, we first split the data into blocks corresponding to codewords of the RS erasure correcting code, encode them and then packetise symbols from the same positions of each codeword. Denoting the length of input as  $l_d$  and the payload size of each packet as  $l_p$ , the minimum number of packets needed can be calculated by  $l_d/l_p$ . Hereafter, a RS block code  $C$  with this scheme is used with the number of information bits  $k = l_p$  and a code rate  $R$ . For a linear block code  $C(n, k)$ , there are  $k$  information packets and  $n - k$  parity-check packets, but the benefit of using the RS codes is that only  $k$  correct packets need to be received and these can be any  $k$  packets. For convenience, the first  $k$  symbols are termed information symbols. The  $i$ -th packet contains  $l_d$  information symbols whose positions are at  $j \cdot k + i$ , where  $j = 0, 1, \dots, (l_d - 1)$  and  $0 \leq i < k$ . At the receiver, all the received information packets need to be restructured into a buffer  $X$  with length  $l_d$ . If the  $i$ -th packet



is received, the symbols contained in the packet should be placed in the buffer at the positions of  $x_{j \times k + i}$ , where  $j = 0, \dots, k - 1$ ; if it has been erased during the transmission, '?'s will be placed in the positions of  $x_{j \times k + i}$ , where  $j = 0, \dots, k - 1$  to mark the symbols as erased.

The packets are transmitted continuously and at the receiver, they are de-packetised and decoded into a buffer with a length of  $l_d$ .

## 6.4 Analysis of Concatenated RS Codes

The most popular decoding algorithm of RS codes is called "error-and-erasure" decoding algorithm, which is preferable to "error-correction-only" decoding algorithm. With this algorithm, an RS code is capable of correcting  $t$  errors and recovering  $\epsilon$  erasures, under the condition of  $2t + \epsilon \leq n - k$ . Then, we can obtain the probability of decoder failure as follows:

$$P_f = \sum_{\epsilon=0}^{n-k} \left\{ \sum_{t=\lfloor (n-k-\epsilon)/2 \rfloor}^{n-\epsilon} P(t|\epsilon)P(\epsilon) \right\} + \sum_{\epsilon=n-k+1}^n P(\epsilon) \quad (6.6)$$

where  $P(\epsilon)$  is the probability of  $\epsilon$  erasures and  $P(t|\epsilon)$  is the conditional probability of  $t$  errors given  $\epsilon$  erasures in the remaining  $n - \epsilon$  positions, which are defined as follows:

$$P(\epsilon) = \binom{n}{\epsilon} \cdot p_{er}^{\epsilon} (1 - p_{er})^{n-\epsilon}. \quad (6.7)$$

and

$$P(t|\epsilon) = \binom{n-\epsilon}{t} \cdot p_e^t (1 - p_e)^{n-\epsilon-t} \quad (6.8)$$

where  $p_{er}$  is the probability of an erasure, and  $p_e$  is the probability of an error but not an erasure.

To ensure that the system can give good performance in the Rayleigh fading channel, the BCH code is used to correct and to detect multiple hard decision errors in each received packet. If the number of errors is small, they are corrected by the Hamming or BCH code. If the number of errors exceeds a threshold, the entire packet is erased and corrected by the RS code. It is well known that a RS code can recover  $n - k$  erasures if a Maximum Likelihood decoding algorithm

## 6. CONCATENATED REED-SOLOMON CODING WITH HARD-DECISION FOR THE RAYLEIGH FADING CHANNEL

---

is implemented. In Chapter 2, we have introduced a complexity-reduced optimal decoding algorithm – the In-place Algorithm. It is always able to solve the maximum number of erasures correctable by the code. However, for a Rayleigh fading channel, we also need to consider the Gaussian noise and fading factor which decrease the energy of each symbol. The system is shown in Figure 6.3.

### 6.5 Numerical Results and Discussion

In this section, we compare RS codes, with different code rates, concatenated with different BCH codes with optimal LDPC codes designed using the Progressive Edge Growth (PEG) technique [32] and bit interleaved for transmission over the Rayleigh fading channel.

First, we evaluated the performance of RS codes with different code rates. As shown in Figure 6.4, the RS code (63, 55, 9) achieved the best performance, as its performance is half order of magnitude better than that of RS(63, 59, 5) code and more than one and a half orders of magnitude better than that of the RS(63, 61, 3) code at a  $FER$  of  $10^{-3}$ .

The simulation results obtained for the RS code (63, 59, 5) concatenated with different BCH codes are given in Figure 6.5. The BCH codes used were the Hamming (63, 57, 3) code, and the (63, 51, 5), (63, 45, 7) and (63, 18, 10) codes respectively. These codes can detect up to a maximum of 2, 4, 6 and 9 errors, respectively or correct up to a maximum of 1, 2, 3 and 4 errors, respectively or a combination of less corrected/detected errors in each received packet. The performance is improved by using more powerful codes until the rate loss causes significant degradation to the  $E_b/N_0$ . The rate loss for the Hamming (63, 57, 3) code is  $-0.43$  dB; for the (63, 51, 5) code, the tradeoff is  $-0.92$  dB; for the (63, 45, 7) code, the tradeoff is  $-1.46$  dB and for the BCH (63, 18, 10) code, the tradeoff is  $-5.44$  dB. Observe that the concatenated codes have different coding gains over the original RS code, especially the one concatenated with the BCH(63, 45, 7), which has a coding gain of approximately 2.2 dB at a  $FER$  of  $10^{-3}$ . Interestingly, the loss is so excessive that when the (63, 18, 10) code is used, the performance is worse than the uncoded performance. Therefore, the rate loss of the inner codes cannot be ignored.

## 6.5 Numerical Results and Discussion

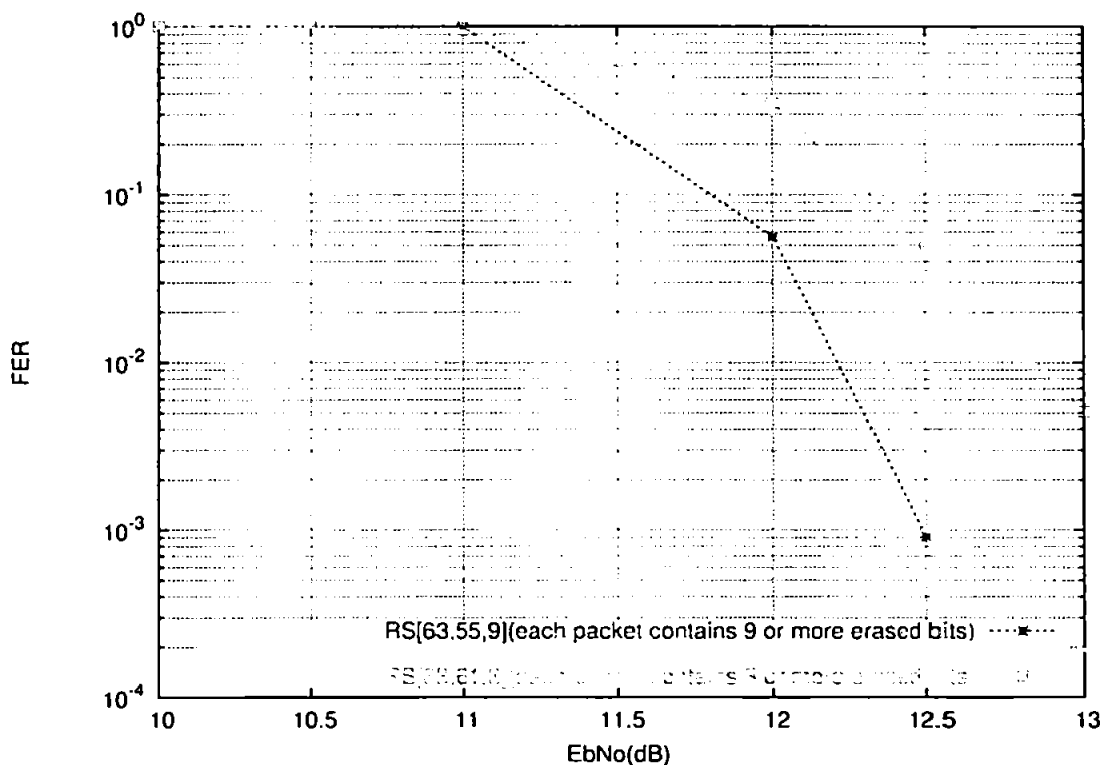


Figure 6.4: RS codes with variable code rates in the Rayleigh fading channel

We also compared the performance between the RS code, concatenated with a BCH code using hard decisions, and the PEG designed LDPC code with soft decision, Belief Propagation, iterative decoding for the Rayleigh fading channel. In this comparison, we applied the RS codes and the LDPC codes with the same code rate and the same packet size. As shown in Figure 6.6, the RS with BCH hard-decision decoder achieved a significant performance improvement over the PEG LDPC soft-decision decoder. At a  $FER$  of  $10^{-3}$ , the RS with BCH hard-decision codes can obtain an average coding gain of 2.0 dB over the LDPC soft-decision code in the Rayleigh fading channel. This is attributable to the RS codes being MDS, optimum codes with maximum likelihood erasure correcting decoding. This factor more than compensates for the loss associated with hard decisions for the Rayleigh fading channel particularly as the LDPC codes are not the most powerful codes due to the necessity for iterative decoding.

## 6. CONCATENATED REED-SOLOMON CODING WITH HARD-DECISION FOR THE RAYLEIGH FADING CHANNEL

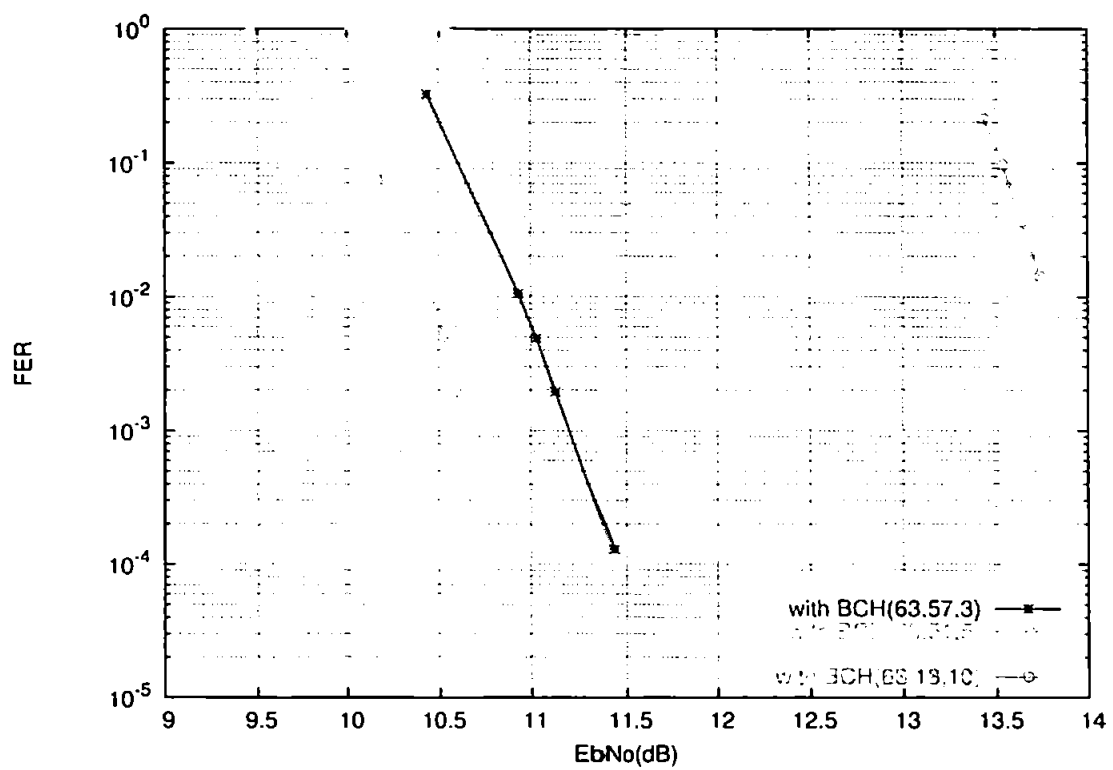


Figure 6.5: RS (63, 59, 5) concatenated by different Hamming or BCH codes in the Rayleigh fading channel

## 6.5 Numerical Results and Discussion

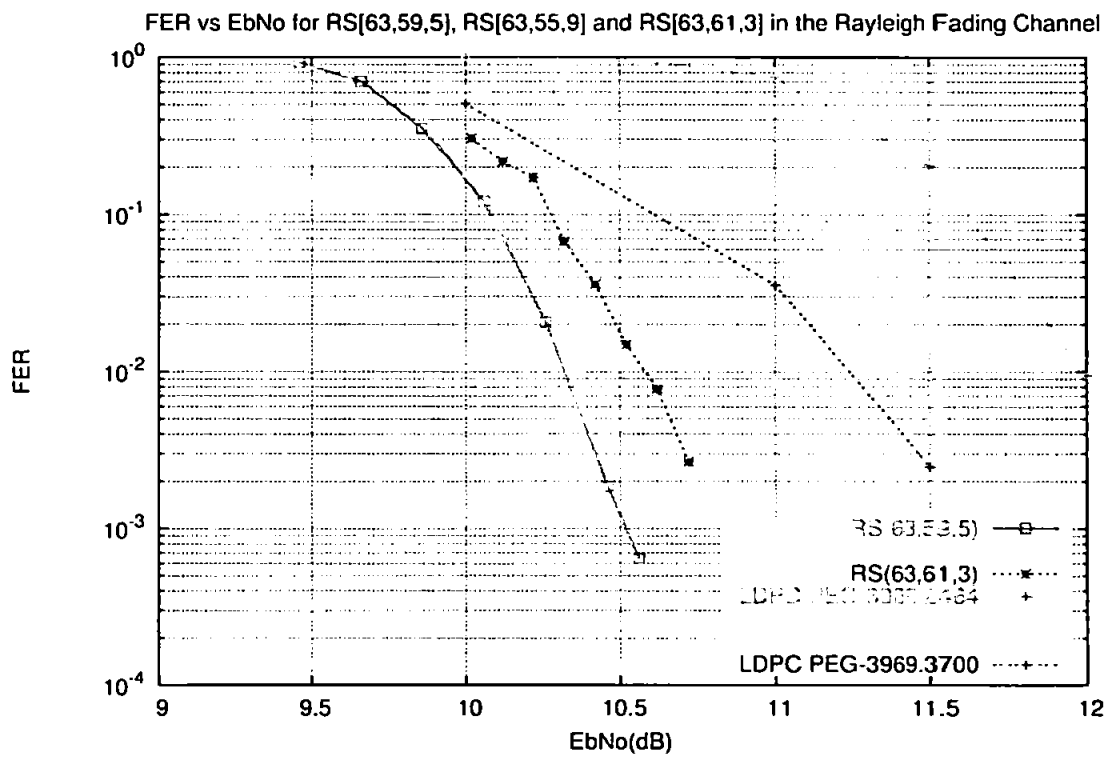


Figure 6.6: RS with BCH hard-decision codes vs. LDPC soft-decision codes in the Rayleigh fading channel

## **6. CONCATENATED REED-SOLOMON CODING WITH HARD-DECISION FOR THE RAYLEIGH FADING CHANNEL**

---

### **6.6 Summary**

In this Chapter, we described the use of RS codes concatenated with BCH codes, with hard-decision decoding for the wireless Rayleigh fading channel. It was shown that the best performance is a function of overall code rate. Furthermore it was shown that the concatenated code combined with simple, hard decision decoding achieves better results than using an optimally designed (PEG) LDPC code combined with soft decision decoding. Further work will provide analysis of the two coding arrangements to show why this is the case. Additionally it will be determined how far the hard decision concatenated arrangement is from capacity for the Rayleigh fading channel.

*I hear and I forget. I see and I remember. I do and I understand.*

Confucius (551 BEC- 479 BEC)

# 7

## Conclusions and Future Work

### 7.1 Conclusions

This thesis is dedicated to the study, design, analysis and evaluation on erasure codes and their decoding algorithms. Primarily, we are interested in the transformation of iterative message-passing algorithms for from the AWGN channel into the BEC.

The first part of the thesis is focused on the decoding algorithms for erasure codes.

Inspired by the BP algorithm and Gallager's bit-flipping algorithm, we have introduced and analysed the matrix-based recovery algorithm and the guess/multi-guess algorithm. The performance of LDPC codes with the recovery algorithm suffers from the error-floor problem. The guess algorithm is capable of reaching an optimal performance when the number of guesses is enough to break the stopping-sets caused by the code structure. However, the computational complexity exponentially increases as the number of guesses is augmented. In search

## 7. CONCLUSIONS AND FUTURE WORK

---

of a MLD for the BEC, we have proposed the In-place algorithm which certainly promises to return an optimal performance and amazingly has the computational complexity less than the conventional quadratic time. (Chapter 2)

In the first part, motivated by the invention of the MLD for the BEC, we have also investigated the ML-approaching algorithms for the AWGN channel. The BESCT algorithm has been introduced and analysed. It combines the merits of the reliability decoding algorithms and the code-structured algorithms. The simulations and analysis have shown that when the information length is fixed, the number of errors in the MRB part is less than 4 and the position of the least likable error is bounded by 4, the BESCT algorithm can obtain the ML performance with a less computational complexity than the OSD algorithm on average. (Chapter 3)

The second part of the thesis is devoted to the application and evaluation of transmission structures to erasure codes.

We have studied the sub-optimal erasure codes including LT codes and Raptor codes in packetised network transmissions. Our investigation and simulation have shown that when the information file is large enough (for example, with more than 10000 packets), LT codes are capable of recovering the information file with the actual number of transmission packets similar to that of information packets. If the information file is as short as the number of information packets less than 5000, LT codes require the transmission packets with the number more than twice than that of information packets. (Chapter 4)

We have devised the product-packetisation structure for packet-data network transmissions. Assisted by their MDS properties, RS codes are capable of realising the desirable rateless transmissions. Combined with the proposed product-packetisation structure, the In-place algorithm at receiver performs the ML performance. Complexity, frame error rate and error statistics are evaluated which shows that the rateless RS product-packetisation coding scheme seems a promising candidate for the application of packet data networks, such as broadcasting and multicasting. (Chapter 5)

We have investigated erasure codes for the Rayleigh fading channel with erasures and errors. We have proposed the concatenated RS coding structured by the product packetisation to recover the erasures and correct the errors. We have



compared the performance of the proposed coding scheme with hard-decision In-place decoding to that of optimally designed (PEG) LDPC code combined with soft-decision iterative decoding. It has shown, at a FER of  $10^{-3}$ , the RS with BCH hard-decision codes can obtain an average coding gain of 2.0  $dB$  over the LDPC soft-decision code in the Rayleigh fading channel. (Chapter 6)

The major contributions of this work include:

1. We have introduced the matrix-based iterative algorithms, which are the recovery algorithm and the guess algorithm. With the matrix representation, we have suggested the efficient way to choose the guesses. Computer simulations show that with a limited number of crucial guesses, the performance of the guess algorithm asymptotically approaches to the ML performance for LDPC codes. For example, for the PEG LDPC (256,128), with the maximum number of crucial guesses of 3, at the erasure probability of 0.35, the performance of the guess algorithm has been improved by the magnitude order of 1.5 compared to that of the recovery algorithm, which is close to the MLD performance by the gap of the magnitude order of 0.5.
2. We have proposed an MLD algorithm, the In-place algorithm, for the BEC. The proposed algorithm focuses on solving the parity-check equations containing erasures, which has its computational complexity less than the conventional quadratic time. It can be applied for both binary codes and non-binary codes.
3. We have proposed a graph-based ML-approaching algorithm for the AWGN channel. We have shown that by applying a bi-directional code tree, the proposed branch-evaluation algorithm based on the cross-correlation metric is equivalent to the classic Dorsch algorithm with a flexible threshold method. The dynamic threshold method speeds up the decoding process when the number of errors in the MRB is less than 4, the position of the least likable error is located within the first 4 less reliable bits in the MRB, compared to the OSD algorithm.

## 7. CONCLUSIONS AND FUTURE WORK

---

4. We have revealed that the popular fountain codes can only approach the sub-optimal performance under the condition of the number of information packets more than 10000.
5. We have proposed the product packetisation of constructing rateless RS codes for use in packetised network transmissions. We show some that efficient rateless RS codes combined with the In-place algorithm at receiver, can achieve the optimal performance with the computational complexity less than  $O(N^{1.5})$ , where  $N$  is the total length of transmitted bits.
6. We have applied the product packetisation structure on concatenated RS coding with the hard-decision In-place algorithm for the Rayleigh fading channel. Computer simulations have shown that the concatenated RS code combined with simple, hard-decision In-place decoding is capable of achieving better performance than using an optimally designed (PEG) LDPC code combined with soft-decision iterative decoding.

## 7.2 Future Work

### 1. *Algebraic RS Decoding Algorithms*

During the study of erasure correcting codes, a huge amount of interest has been raised on RS codes. The non-binary structure of RS codes, which has been considered as the reason of the high encoding/decoding complexity, becomes one of the drawbacks of these powerful codes. The breakthrough of the Guruswami-Sudan algorithm has stirred up the research focused on the algebraic decoding algorithms for RS codes. It will be valuable to further study these decoding algorithms and devise practical structures for different applications.

### 2. *Implementation of Rateless RS Product-packetisation Structure*

To realise the core findings of our research work on the In-place algorithm and the product-packetisation structure into the practical world is absolutely meaningful. A field-programmable gate array (FPGA) board is a popular integrated-circuit, especially deployed in the DSP/Communications systems. The implementation of rateless RS product-packetisation structure requires a further understanding of an efficient FPGA programming and the limited memory allocation.

### 3. *Further Study on the Random Matrix Theory*

After revealing the limitations of fountain codes, the theory of random matrix rings the bell. To devise a fountain code beyond its limitation on the short information file, it is necessary to further understand the random matrix theory.

## 7. CONCLUSIONS AND FUTURE WORK

---

## References

- [1] 3GPP (2005). *Technical Specification Group Services and System Aspects; Multimedia Broadcast/Multicast Service; Protocols and Codecs*. 3GPP TS 26.346 V6.1.0. 103, 108
- [2] AMBROZE, M., TOMLINSON, M. & YANG, L. (2009). Exhaustive weight spectrum analysis of some well known ldpc codes. In *the 10<sup>th</sup> IEEE International Communication Theory and Applications (ISCTA 09)*, Lake District, UK. xi, 62, 76, 77, 78
- [3] BELLORADO, J. & KAVCIC, A. (2006). A low-complexity method for chase-type decoding of reed-solomon codes. In *in Proc. IEEE Int. Symp. Information Theory*. 11
- [4] BERLEKAMP, E.R. (1968). *Algebraic Coding Theory*. New York: McGraw-Hill. 3, 9, 10, 80, 130
- [5] BERLEKAMP, E.R., MCELIECE, R. & VAN TILBORG, H. (1978). On the inherent intractability of certain coding problems. *IEEE Transactions on Information Theory*, **24**, 384–386. 23
- [6] BERROU, C., GLAVIEUX, A. & THITIMAJSHIMA, P. (1993). Near shannon limit error-correcting coding: Turbo codes. In *Proc. IEEE International Conference on Communications*, 1064–1070, Geneva, Switzerland. 4
- [7] BLEICHENBACHER, D., KLYAYIAS, A. & YUNG, M. (2003). Cdecoding of interleaved reed-solomon codes over noisy data. In *Proc. of ICALP 2003*, 97–108. 114

## REFERENCES

---

- [8] BOSE, R.C. & RAY-CHAUDHURI, D.K. (1960). On a class of error-correcting binary group codes. *Inform. Contr.*, 3, 68–79. 3
- [9] CAI, J. (2009). *A Design Note on the BESCT Algorithm*. Personal Study Notes. 17
- [10] CAI, J., TJHAI, C., TOMLINSON, M., AMBROZE, M. & AHMED, M.Z. (2005). An efficient solution to packet loss: Erasure correcting codes. In *Proc. 4<sup>th</sup> IASTED International Conference CSN*, 224–229. 16, 17, 35, 121
- [11] CAI, J., TOMLINSON, M., AMBROZE, M. & AHMED, M. (2006). Advances in iterative decoding and maximum likelihood decoding for the packet network with comparisons to fountain codes over the erasure channel. In *in Proc. 4<sup>th</sup> International Symposium on Turbo Codes in connection with 6<sup>th</sup> International ITG-Conference on Source and Channel Coding*, Munich, Germany. 18
- [12] CAI, J., TOMLINSON, M., AMBROZE, M. & AHMED, M. (2006). Comparison of concatenated reed-solomon coding with hard-decision to soft-decision ldpc coding for the rayleigh fading channel. In *in Proc. IEEE Information Theory Workshop*, Chengdu, China. 18
- [13] CHASE, D. (1972). A class of algorithms for decoding block codes with channel measurement information. *IEEE Transactions on Information Theory*, IT-18, 170–182. 3
- [14] CHIEN, R. (1964). Cyclic decoding procedures for bose-chaudhuri-hocquenghem codes. *IEEE Transactions on Information Theory*, 10, 357–363. 10
- [15] COPPERSMITH, D. & WINOGRAD, S. (1987). Matrix multiplication via arithmetic progressions. In *Proc. 19<sup>th</sup> ACM Symp. Theory of Computing*, 1–6. 41
- [16] DI, C., PROIETTI, D., TELATAR, I.E., RICHARDSON, T. & URBANKE, R. (2002). Finite-length analysis of low-density parity-check codes on the binary

## REFERENCES

---

- erasure channel. *IEEE Transactions on Information Theory*, **48**, 1570–1579. 15, 25, 31, 35, 114, 120
- [17] DORSCH, B. (1974). A decoding algorithm for binary block codes and  $j$ -ary output channels. *IEEE Transactions on Information Theory*, **20**, 391–394. 62
- [18] ELIAS, P. (1954). Error-free coding. *IRE Trans.*, **PGIT-4**, 29–37. 23
- [19] ELIAS, P. (1955). Coding for noisy channels. *IRE Conv. Rec.*, **4**, 37–46. 3, 23, 113
- [20] ELIAS, P. (1956). Coding for two noisy channels. In *Third London Symposium on Information Theory*, Academic Press, New York. 8
- [21] ELIAS, P. (1957). List decoding for noisy channels. *IRE Trans.*, **PGIT-4**, 37–47. 10
- [22] FORNEY, G.D. (1965). On decoding bch codes. *IEEE Transactions on Information Theory*, **11**, 549–557. 10
- [23] FOSSORIER, M. & LIN, S. (1995). Soft-decision decoding of linear block codes based on ordered statistics. *IEEE Transactions on Information Theory*, **41**, 1379–1396. 61, 63, 65
- [24] GALLAGER, R. (1963). *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press. 3, 15, 24, 114
- [25] GOFF, S.L., GLAVIEUX, A. & BERROU, C. (1994). Turbo-codes and high spectral efficiency modulation. In *Proc. IEEE Int. Conf. Communications*, 645–649. 129
- [26] GURUSWAMI, V. & SUDAN, M. (1999). Improved decoding of reed-solomon codes and algebraic geometry codes. *IEEE Transactions on Information Theory*, **45**, 1757–1767. 10
- [27] HALL, E.K. & WILSON, S.G. (1998). Design and analysis of turbo codes on rayleigh fading channels. *IEEE J. Select. Areas Commun.*, 160–174. 129

## REFERENCES

---

- [28] HAMMING, R. (1950). Error detecting and error correcting codes. *Bell Syst. Tech. J.*, **29**, 41-56. 2
- [29] HARTLEY, R. (1928). Transmission of information. *Bell Syst. Tech. J.*, **7**, 535-563. 1
- [30] HOCQUENGHEM, A. (1959). Codes correcteurs d'erreurs. *Chiffres*, **2**, 147-156. 3
- [31] HOU, J., SIEGEL, P.H. & MILSTEIN, L.B. (2001). Performance analysis and code optimisation of low density parity-check codes on rayleigh fading channels. *IEEE J. Select. Areas Commun.*, **19**, 924-934. 129
- [32] HU, X., ELEFThERIOU, E. & ARNOLD, D.M. (2005). Regular and irregular progressive edge-growth Tanner graphs. *IEEE Transactions on Information Theory*, **51**, 386-398. 33, 77, 136
- [33] JUNG, P. (1995). Novel low complexity decoder for turbo codes. *Electron. Lett.*, 86-87. 129
- [34] KOETTER, R. & VARDY, A. (2003). Algebraic soft-decision decoding of reed-solomon codes. *IEEE Transactions on Information Theory*, 2809-2825. 10
- [35] KOU, Y., LIN, S. & FOSSORIER, M. (2001). Low density parity check codes based on finite geometries: rediscovery and new results. *IEEE Transactions on Information Theory*, **47**, 2711-2736. 51
- [36] LEE, S.W. & KUMAR, B.V.K.V. (2008). Soft-decision decoding of reed-solomon codes using successive error-and-erasure decoding. In *Proc. IEEE Global Telecom. Conf.* 11
- [37] LIN, S. & COSTELLO, JR., D.J. (2004). *Error Control Coding: Fundamentals and Applications*. Pearson Education, Inc, 2nd edn., ISBN 0 13 017973 6. 6, 44, 61
- [38] LUBY, M. (2002). Lt-code. In *Proc. 43rd Annual IEEE Symposium on the Foundations of Computer Science*, 271-280. 11, 48, 89, 93, 95, 114



## REFERENCES

---

- [39] LUBY, M., MITZENMACHER, M., SHOKROLLAHI, M. & SPIELMAN, D. (2001). Efficient erasure correcting codes. *IEEE Transactions on Information Theory*, **47**, 569–584. 24, 114
- [40] LUBY, M., MATSON, M., GASIBLE, T., STOCKHAMMER, T. & XU, W. (2006). Raptor codes for reliable download delivery in wireless broadcast systems. In *Proc. Consumer Communications and Networking Conference*. 100
- [41] MACKAY, D.J.C. (2003). Information Theory, Inference and Learning Algorithms. Available at: <http://www.inference.phy.cam.ac.uk/itprnn/book.pdf>. 89, 103
- [42] MACKAY, D.J.C. & NEAL, R.M. (1996). Nears Shannon limit performance of low-density parity-check codes. *Electronic Letter*, **32**. 4
- [43] MACWILLIAMS, F.J. & SLOANE, N.J.A. (1977). *The Theory of Error-Correcting Codes*. North-Holland. 6, 9, 56, 115, 118, 119, 130
- [44] MASSEY, J.L. (1963). *Threshold Decoding*. MIT Press, Cambridge, MA, United States of America. 4, 9, 24
- [45] MASSEY, J.L. (1969). Shift-register synthesis and BCH decoding. *IEEE Transactions on Information Theory*, **IT-15**, 122–127. 3, 10
- [46] MULLER, D.E. (1954). Application of Boolean algebra to switching circuit design and to error detection. *IRE Trans. Electron. Comput.*, **EC-3**, 6–12. 3
- [47] NYQUIST, H. (1924). Certain factors affecting telegraph speed. *Bell Syst. Tech. J.*, **3**, 324–346. 1
- [48] ODLYZKO, A.M. (1985). Discrete logarithms in finite fields and their cryptographic significance. In *Advances in Cryptology: Proceeding of Eurocrypt '84*, 224–314. 41

## REFERENCES

---

- [49] OSWALD, P. & SHOROLLAH, M.A. (2002). Capacity-achieving sequences for the erasure channel. *IEEE Transactions on Information Theory*, **48**, 3019–3028. 114
- [50] PAPOULIS, A. (1965). *Probability, random variables and stochastic processes*. McGraw-Hill. 130
- [51] PEARL, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA. 24
- [52] PETERSON, W.W. (1960). Encoding and error-correction procedures for the bose-chaudhuri codes. *IRE Trans. Inform. Theory*, **IT-6**, 459–470. 3
- [53] PETERSON, W.W. (1961). *Error Correcting Codes*. The MIT Press. 3, 49
- [54] PISHRO-NIK, H. & FEKRI, F. (2004). On decoding of low-density parity-check codes over the binary erasure channel. *IEEE Transactions on Information Theory*, **20**, 439–454. 114
- [55] PROAKIS, J.G. (2001). *Digital Communications*. McGraw-Hill, New York, 4th edn. 131, 133
- [56] RAINS, E. & SLOANE, N.J.A. (1998). *Self-dual Codes*. V.S. Pless and W.C.Huffman. 81
- [57] REED, I. & SOLOMON, G. (1960). Polynomial codes over certain finite fields. *J. Soc. Indust. Appl. Math*, **8**, 300–304. 9
- [58] REED, I.S. (1954). A class of multiple-error-correcting codes and the decoding scheme. *IRE Trans. Inform. Theory*, **IT-4**, 38–49. 3
- [59] REED, I.S. & SOLOMON, G. (1960). Polynomial codes over certain finite fields. *J. Siam*, **8**, 300–304. 3
- [60] RICHARDSON, T. & URBANKE, R. (2001). Efficient encoding of low density parity check codes. *IEEE Transactions on Information Theory*, 638–656. 27, 41

- 
- [61] RIZZO, L. (1997). Effective erasure codes for reliable computer communication protocols. In *Proc. ACM SIGCOMM 1997*, 24–36. 8
- [62] ROSNES, E. & O.YTREHUS (2005). Improved algorithms for the determination of turbo-code weight distributions. *IEEE Transactions on Communications*, 62
- [63] ROSNES, E. & YTREHUS, O. (2006). Turbo decoding on the binary erasure channel: finite-length analysis and turbo stopping sets. In *International Symposium on Information Theory*. 62
- [64] ROSNES, E. & YTREHUS, O. (2007). An algorithm to find all small-size stopping sets of low-density parity-check matrices. In *International Symposium on Information Theory*, Nice, France. 62
- [65] ROTH, R.M. & RUCKENSTEIN, G. (2000). Efficient decoding of reed-solomon codes beyond half the minimum distance. *IEEE Transactions on Information Theory*, 46, 246–257. 130
- [66] SHANNON, C.E. (1948). A mathematical theory of communication. *Bell Syst. Tech. J.*, 27, 379–423. 2
- [67] SHOKROLLAHI, A. (2000). Capacity-achieving sequences. *IMA Volumes in Mathematics and its Applications*, 123, 153–166. 114
- [68] SHOKROLLAHI, A. (2003). *Raptor codes*. Digital Fountain. 12, 90, 101, 103, 114
- [69] SHOKROLLAHI, A. (2006). Raptor codes. *IEEE Transactions on Information Theory*, 52, 2551–2567. xi, 96, 97, 98, 99
- [70] SILVERMAN, R.A. & BALSER, M. (1954). Coding for constant-data-rate systems. *IRE Trans. Inform. Theory*, PGIT-4, 50–63. 3
- [71] STRASSEN, V. (1969). Gaussian elimination is not optimal. *Numerische Math.*, 13, 354–356. 41

- [72] SUDAN, M. (1997). Decoding of reed solomon codes beyond the error-correction bound. *Journal of Complexity*, 13, 180–193. 130
- [73] TJHAI, C., TOMLINSON, M., AMBROZE, M. & AHMED, M. (2005). Cyclotomic idempotent-based binary cyclic codes. *Electron. Lett.*, 41, 341–343. 51, 78
- [74] TOMLINSON, M., CAI, J., TJHAI, C., AMBROZE, M. & AHMED, M.Z. (2004). *System for correcting deleted or missing symbols*. U.K. Patent application No. 0428042.6. 16, 18, 121
- [75] TOMLINSON, M., TJHAI, C. & AMBROZE, M. (2007). Extending the dorsch decoder towards achieving maximum-likelihood decoding for linear codes. *IET Communications*, 1, 479–488. 62
- [76] TOMLINSON, M., TJHAI, C., CAI, J. & AMBROZE, M. (2007). Analysis of the distribution of the number of erasures correctable by a binary linear code and the link to low weight codewords. *IET Proceedings Communications*, 539–548. 16
- [77] VITERBI, A.J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, IT-13, 260–269. 3
- [78] WIKIPEDIA (2009). Fountain Code. Available at [http://en.wikipedia.org/wiki/Fountain\\_code](http://en.wikipedia.org/wiki/Fountain_code). 90
- [79] WOZENCRAFT, J.M. & REIFFEN, B. (1961). *Sequential Decoding*. MIT Press, Cambridge, MA, United States of America. 3
- [80] XIA, H. & CRUZ, J.R. (2007). Reliability-based forward recursive algorithms for algebraic soft-decision decoding of reed-solomon codes. In *IEEE Commun. Letters*. 11

# A

## Publications

The published papers and patent are listed in the followings:

- Tomlinson, M., Tjhai, C., Cai, J. and Ambroze, M., "Analysis of the Distribution of the Number of Erasures Correctable by a Binary Linear Code and the Link to Low Weight Codewords", *IET Proceedings Communications* 1(3):539-548, June 2007.
- Cai, J., Tomlinson, M., Tjhai, C., Ambroze, M., Ahmed, M., "Comparison of Concatenated Reed-Solomon Coding with Hard-decision to Soft-decision LDPC Coding for the Rayleigh Fading Channel", *in Proc. IEEE Information Theory Workshop, Chengdu, China, 22-26 Oct., 2006.*
- Cai, J., Tomlinson, M., Tjhai, C., Ambroze, M. and Ahmed, M., "Advances in Iterative Decoding and Maximum Likelihood Decoding for the Packet Network with Comparisons to Fountain Codes over the Erasure Channel", *in Proc. 4th International Symposium on Turbo Codes in connection with*

6th International ITG-Conference on Source and Channel Coding, Munich, Germany, 3-7 April, 2006.

- *Cai, J.*, Tjhai, C., Tomlinson, M., Ambroze, M. and Ahmed, M., "An Efficient Solution to Packet Loss: Erasure Correcting Codes", in Proc. 4th IASTED International Conference Communication Systems and Networks, Benidorm, Spain, 12-14 Sep., 2005.
- *Cai, J.*, Ambroze, M., "Analysis on Decoding Algorithms Based on Sectionalised Trellises of Block Codes and Their Dual", in Proc. PREP, Lancaster, 30 Mar-1 April, 2005.
- Tomlinson, M., *Cai, J.*, Tjhai, C., Ambroze, M. and Ahmed, M., "System for Correcting Deleted or Missing Symbols", UK Patent Application, GB 2421599, 28 June, 2006.

# Analysis of the distribution of the number of erasures correctable by a binary linear code and the link to low-weight codewords

M. Tomlinson, C. Tjhai, J. Cai and M. Ambroze

**Abstract:** The number and weight of low-weight codewords of a binary linear code determine the erasure channel performance. Analysis is given of the probability density function of the number of erasures correctable by the code in terms of the weight enumerator polynomial. For finite-length codes, zero erasure decoder error rate is impossible, even with maximum-distance-separable (MDS) codes and maximum-likelihood decoding. However, for codes that have binomial weight spectra, for example BCH, Goppa and double-circulant codes, the erasure correction performance is close to that of MDS codes. One surprising result is that, for many  $(n, k)$  codes, the average number of correctable erasures is almost equal to  $n - k$ , which is significantly larger than  $d_{\min} - 1$ . For the class of iteratively decodable codes (LDPC and turbo codes), the erasure performance is poor in comparison to algebraic codes designed for maximum  $d_{\min}$ . It is also shown that the turbo codes that have optimised  $d_{\min}$  have significantly better performance than LDPC codes. A probabilistic method, which has considerably smaller search space than that of the generator matrix-based methods, is presented to determine the  $d_{\min}$  of a linear code using random erasure patterns. Using this approach, it is shown that there are (168, 84, 24) and (216, 108, 24) quadratic double-circulant codes.

## 1 Introduction

The erasure correcting performance of codes and associated decoders has received renewed interest in the study of coding over packet networks as a means of providing efficient computer communication protocols [1]. In [2, 3], it is shown that  $(n, k)$  erasure correcting codes can be used as an efficient and effective solution to recover lost packets in communication networks without the need of retransmission. This is achieved by arranging data in a squared-matrix representation (product-packetisation), where each column contains a codeword and each row represents a packet to be transmitted across the network. This product-packetisation structure allows efficient communications as not all packets need to be transmitted; the missing packets are treated as erasures and a decoder can reconstruct them provided that  $k$  independent packets are successfully received. This efficient communication scheme has been further improved in [4] to include error correcting codes in addition to erasure correcting codes. The erasure correcting codes are also used in distributed computing systems. For example, Rabin [5] proposed a scheme to use an  $(n, k)$  erasure correcting code to improve data availability in distributed storage system. The data are divided into  $k$  blocks and encoded to produce  $n$  blocks, which are distributed across  $n$  hosts. In this way, data can still be recovered despite the unavailability of some hosts. In addition to those mentioned above, the erasure performance of codes, in

particular LDPC codes are used as a measure of predicting the code performance for the additive white gaussian noise channel [6, 7].

It is well known that an  $(n, k, d_{\min})$  error correcting code  $C$ , where  $n$  and  $k$  denote the code length and information length, respectively, can correct up to  $d_{\min} - 1$  erasures [8, 9] where  $d_{\min}$  is the minimum Hamming distance of the code. However, it is not so well known that the average number of erasures correctable by most codes is significantly higher than this and almost equal to  $n - k$ . One of the earlier analyses of the erasure correction performance of particular linear block codes is provided in a key-note paper by Dumer and Farrell [10], who derived the erasure correcting performance of long binary BCH codes and their duals and showed that these codes achieved capacity for the erasure channel.

In this paper, an expression is obtained for the probability density function (PDF) of the number of correctable erasures as a function of the weight enumerator function of the linear code. Analysis results of several common codes are given in comparison to maximum likelihood (ML) decoding performance for the binary erasure channel. Many codes including BCH codes, Goppa codes, double circulant, self-dual codes have weight distributions that closely match the binomial distribution [9, 11–13]. It is shown for these codes that a lower bound of the average number of correctable erasures is  $n - k - 2$ . The decoder error rate performance for these codes is also analysed. Results are given for rate 0.9 codes and it is shown that for code lengths of 5000 bits or longer there is insignificant difference in performance between these codes and the theoretical optimum maximum distance separable (MDS) codes. The results for specific codes are given including BCH codes, extended quadratic residue codes, LDPC codes designed using the progressive edge growth (PEG) technique [14] and turbo codes [15].

© The Institution of Engineering and Technology 2007

doi:10.1049/iet-com:20060393

Paper first received 4th July and in revised form 5th November 2006

The authors are with the Fixed and Mobile Communications Research, University of Plymouth, Plymouth PL4 8AA, UK

E-mail: ctjhai@plymouth.ac.uk

*IET Commun.*, 2007, 1, (3), pp. 539–548

539

## 2 Derivation of the PDF of correctable erasures

### 2.1 Background and definitions

A set of  $s$  erasures is a list of erased bit positions defined as  $f_i$  where  $0 < i < s$  and  $f_i \in \{0 \dots n-1\}$ . A codeword  $x = (x_0, x_1, \dots, x_{n-1})$  satisfies the parity check equations of the parity check matrix  $H$ , that is  $Hx^T = 0$ .

A codeword with  $s$  erasures is defined as  $x = (x_{u_0}, x_{u_1}, \dots, x_{u_{s-1}}, x_{f_0}, x_{f_1}, \dots, x_{f_{s-1}})$ , where  $x_{u_i}$  are the unerased co-ordinates of the codeword and the set of  $s$  erased co-ordinates is defined as  $f_s$ .

There are a total of  $n - k$  parity check equations, provided the erased bit positions correspond to independent columns of the  $H$  matrix; each of the erased bits may be solved using a parity check equation derived by the classic technique of Gaussian reduction [6, 8, 9]. For the MDS codes [9], any set of  $s$  erasures are correctable by the code provided that

$$s \leq n - k \quad (1)$$

Unfortunately, the only MDS binary codes are trivial [9].

### 2.2 Correspondence between uncorrectable erasure patterns and low-weight codewords

Provided the code is capable of correcting the set of  $s$  erasures, then a parity check equation may be used to solve each erasure, viz

$$\begin{aligned} x_{f_0} &= h_{0,0}x_{u_0} + h_{0,1}x_{u_1} + h_{0,2}x_{u_2} + \dots + h_{0,n-s-1}x_{u_{n-s-1}} \\ x_{f_1} &= h_{1,0}x_{u_0} + h_{1,1}x_{u_1} + h_{1,2}x_{u_2} + \dots + h_{1,n-s-1}x_{u_{n-s-1}} \\ x_{f_2} &= h_{2,0}x_{u_0} + h_{2,1}x_{u_1} + h_{2,2}x_{u_2} + \dots + h_{2,n-s-1}x_{u_{n-s-1}} \\ &\vdots \\ x_{f_{s-1}} &= h_{s-1,0}x_{u_0} + h_{s-1,1}x_{u_1} + h_{s-1,2}x_{u_2} \\ &\quad + \dots + h_{s-1,n-s-1}x_{u_{n-s-1}} \end{aligned}$$

where  $h_{ij}$  is the coefficient of row  $i$  and column  $j$  of  $H$ .

As the parity check equations are Gaussian reduced, no erased bit is a function of any other erased bits. There will also be  $n - k - s$  remaining parity check equations, which do not contain the erased bit co-ordinates  $x_{f_i}$ , namely

$$\begin{aligned} h_{s,0}x_{u_0} + h_{s,1}x_{u_1} + h_{s,2}x_{u_2} + \dots + h_{s,n-s-1}x_{u_{n-s-1}} &= 0 \\ h_{s+1,0}x_{u_0} + h_{s+1,1}x_{u_1} + h_{s+1,2}x_{u_2} + \dots + h_{s+1,n-s-1}x_{u_{n-s-1}} &= 0 \\ h_{s+2,0}x_{u_0} + h_{s+2,1}x_{u_1} + h_{s+2,2}x_{u_2} + \dots + h_{s+2,n-s-1}x_{u_{n-s-1}} &= 0 \\ &\vdots \\ h_{n-k-1,0}x_{u_0} + h_{n-k-1,1}x_{u_1} + h_{n-k-1,2}x_{u_2} \\ &\quad + \dots + h_{n-k-1,n-s-1}x_{u_{n-s-1}} = 0 \end{aligned}$$

Further to this, the hypothetical case is considered where there is an additional erased bit  $x_{f_s}$ . This bit co-ordinate is clearly equal to one of the previously unerased bit co-ordinates, denoted as  $x_{u_p}$ .

$$x_{f_s} = x_{u_p}$$

Also, in this case it is considered that these  $s + 1$  erased co-ordinates do not correspond to  $s + 1$  independent columns of the  $H$  matrix, but only to  $s + 1$  dependent columns. This means that  $x_{u_p}$  is not contained in any of the  $n - k - s$  remaining parity check equations, and cannot be solved as the additional erased bit.

For the first  $s$  erased bits whose co-ordinates do correspond to  $s$  independent columns of the  $H$  matrix, the set of codewords is considered in which all of the unerased co-ordinates are equal to zero except for  $x_{u_p}$ . In this case, the parity check equations above are simplified to

$$\begin{aligned} x_{f_0} &= h_{0,p}x_{u_p} \\ x_{f_1} &= h_{1,p}x_{u_p} \\ x_{f_2} &= h_{2,p}x_{u_p} \\ &\vdots \\ x_{f_{s-1}} &= h_{s-1,p}x_{u_p} \end{aligned} \quad (2)$$

By definition, as there are, at least  $n - s - 1$  zero co-ordinates contained in each codeword, the maximum weight of any of the codewords above is  $s + 1$ . Furthermore, any erased co-ordinate that is zero may be considered as an unsolved co-ordinate, since no non-zero co-ordinate is a function of this co-ordinate. This leads to the following theorem.

**Theorem 1:** The non-zero co-ordinates of a codeword of weight  $w$  that is not the juxtaposition of two or more lower weight codewords, provide the co-ordinate positions of  $w - 1$  erasures that can be solved and provide the co-ordinate positions of  $w$  erasures that cannot be solved.

**Proof:** The co-ordinates of a codeword of weight  $w$  must satisfy the equations of the parity check matrix. With the condition that the codeword is not constructed from the juxtaposition of two or more lower weight codewords, the codeword must have  $w - 1$  co-ordinates that correspond to linearly independent columns of the  $H$  matrix and  $w$  co-ordinates that correspond to linearly dependent columns of the  $H$  matrix.  $\square$

**Corollary 1:** Given  $s$  co-ordinates corresponding to an erasure pattern containing  $s$  erasures,  $s \leq (n - k)$ , of which  $w$  co-ordinates are equal to the non-zero co-ordinates of a single codeword of weight  $w$ , the maximum number of erasures that can be corrected is  $s - 1$  and the minimum number that can be corrected is  $w - 1$ .

**Corollary 2:** Given  $w - 1$  co-ordinates that correspond to linearly independent columns of the  $H$  matrix and  $w$  co-ordinates that correspond to linearly dependent columns of the  $H$  matrix a codeword can be derived that has a weight less than or equal to  $w$ .

The weight enumeration function of a [9] is usually described as a homogeneous polynomial of degree  $n$  in  $x$  and  $y$ .

$$W_C(x, y) = \sum_{i=0}^{n-1} A_i x^{n-i} y^i$$

The support of a codeword is defined [9] as the co-ordinates of the codeword that are non-zero. The probability of the successful erasure correction of  $s$  or more erasures is equal to the probability that no subset of the  $s$  erasure co-ordinates corresponds to the support of any codeword.



The number of possible erasure patterns of  $s$  erasures of a code of length  $n$  is  $\binom{n}{s}$ . For a single codeword of weight  $w$ , the number of erasure patterns with  $s$  co-ordinates that include the support of this codeword is  $\binom{n-w}{s-w}$ . Thus, the probability of a subset of the  $s$  co-ordinates coinciding with the support of a single codeword of weight  $w$ ,  $\text{Prob}(x_w \in f_s)$  is given by

$$\text{Prob}(x_w \in f_s) = \frac{\binom{n-w}{s-w}}{\binom{n}{s}}$$

and

$$\text{Prob}(x_w \in f_s) = \frac{(n-w)!s!(n-s)!}{n!(s-w)!(n-s)!}$$

simplifying

$$\text{Prob}(x_w \in f_s) = \frac{(n-w)!s!}{n!(s-w)!}$$

By assuming that no erasure pattern includes more than one codeword, a lower bound of the probability of successful erasure correction of  $s$  or more erasures may be derived from the probability that the erasure pattern does contain a codeword of weight  $w \leq s$

$$\text{Prob}(s) = \sum_{j=d_{\min}}^s A_j \frac{(n-j)!s!}{n!(s-j)!}$$

The probability of the code being able to correct exactly  $s$  erasures, but no more,  $\text{Pr}(s)$  for  $s < n-k$  is

$$\text{Pr}(s) = \text{Prob}(s+1) - \text{Prob}(s) \quad (3)$$

and

$$\text{Pr}(s) = \sum_{j=d_{\min}}^{s+1} A_j \frac{(n-j)!(s+1)!}{n!(s+1-j)!} - \sum_{j=d_{\min}}^s A_j \frac{(n-j)!s!}{n!(s-j)!} \quad (4)$$

for  $s = n-k$ ,  $\text{Prob}(n-k+1) = 1$  and

$$\text{Pr}(n-k) = 1 - \sum_{j=d_{\min}}^{n-k} A_j \frac{(n-j)!(n-k)!}{n!(n-k-j)!} \quad (5)$$

The average number of erasures corrected by the code  $N_{\text{erase}}$ , is given by

$$N_{\text{erase}} = \sum_{s=1}^{n-k} s(\text{Prob}(s+1) - \text{Prob}(s))$$

Carrying out the sum in reverse order, the equation simplifies to

$$N_{\text{erase}} = (n-k)\text{Prob}(n-k+1) - \sum_{s=1}^{n-k} \text{Prob}(s)$$

Noting that there is a probability 1, that  $(n-k+1)$  erasures cannot be corrected,  $N_{\text{erase}}$  and  $\text{Prob}(s) = 0$  for  $s < d_{\min}$ ,  $N_{\text{erase}}$  becomes

$$N_{\text{erase}} = (n-k) - \sum_{s=d_{\min}}^{n-k} \text{Prob}(s)$$

Substituting  $\text{Prob}(s)$  gives

$$N_{\text{erase}} = (n-k) - \sum_{s=d_{\min}}^{n-k} \sum_{j=d_{\min}}^s A_j \frac{(n-j)!s!}{n!(s-j)!} \quad (6)$$

The terms responsible for the shortfall in performance compared to an MDS code,  $\text{MDS}_{\text{shortfall}}$  is evident from (6)

$$\text{MDS}_{\text{shortfall}} = \sum_{s=d_{\min}}^{n-k} \sum_{j=d_{\min}}^s A_j \frac{(n-j)!s!}{n!(s-j)!} \quad (7)$$

As well as determining the performance shortfall, compared to MDS codes, in terms of the number of correctable erasures it is also possible to determine the loss from capacity for the erasure channel. The capacity of the erasure channel with erasure probability  $p$  was originally determined by Elias [16] to be  $1-p$ . The capacity may be approached with zero codeword error for very long codes, even using non-MDS codes such as BCH codes [10]. However, short codes, even MDS codes will produce a non-zero frame error rate (FER). For  $(n, k, n-k+1)$  MDS codes, a codeword decoder error is deemed to occur whenever there are more than  $n-k$  erasures. (It is assumed here that the decoder does not resort to guessing erasures that cannot be solved.) This probability,  $P_{\text{MDS}}(p)$  is given by

$$P_{\text{MDS}}(p) = 1 - \sum_{s=0}^{n-k} \frac{n!}{(n-s)!s!} p^s (1-p)^{(n-s)} \quad (8)$$

The probability of codeword decoder error for the code may be derived from the weight enumerator of the code by using (4).

$$P_{\text{code}}(p) = \sum_{s=d_{\min}}^{n-k} \sum_{j=d_{\min}}^s A_j \frac{(n-j)!s!}{n!(s-j)!(n-s)!s!} p^s (1-p)^{(n-s)} + \sum_{s=n-k+1}^n \frac{n!}{(n-s)!s!} p^s (1-p)^{(n-s)} \quad (9)$$

which simplifies to

$$P_{\text{code}}(p) = \sum_{s=d_{\min}}^{n-k} \sum_{j=d_{\min}}^s A_j \frac{(n-j)!(n-s)!}{(s-j)!} \times p^s (1-p)^{(n-s)} + P_{\text{MDS}}(p) \quad (10)$$

The first term in the above equation represents the loss from MDS code performance.

### 3 Codes whose weight enumerator coefficients are approximately binomial

It is well known that the distance distribution for many linear, binary codes including BCH codes, Goppa codes, self-dual codes [9, 11–13] approximates to a binomial distribution. Accordingly

$$A_j \simeq \frac{n!}{(n-j)!j!2^{n-k}} \quad (11)$$

For these codes, for which the approximation is true, the shortfall in performance compared to an MDS code,

MDS<sub>shortfall</sub> is obtained by substitution into (7)

$$\text{MDS}_{\text{shortfall}} = \sum_{s=1}^{n-k} \sum_{j=1}^s \frac{n!}{(n-j)!j!2^{n-k}} \frac{(n-j)!s!}{n!(s-j)!} \quad (12)$$

which simplifies to

$$\text{MDS}_{\text{shortfall}} = \sum_{s=1}^{n-k} \frac{2^s - 1}{2^{n-k}} \quad (13)$$

which leads to the simple result

$$\text{MDS}_{\text{shortfall}} = 2 - \frac{n-k-2}{2^{n-k}} \simeq 2 \quad (14)$$

It is apparent that for these codes the MDS shortfall is just two bits from correcting all  $n - k$  erasures. It is shown later using the actual weight enumerator functions for codes, where these are known, that this result is slightly pessimistic since in the above analysis there is a non-zero number of codewords with distance less than  $d_{\min}$ . However, the error attributable to this is quite small. Simulation results for these codes show that the actual MDS shortfall is closer to 1.6 bits due to the assumption that there is never an erasure pattern, which has the support of more than one codeword.

For these codes whose weight enumerator coefficients are approximately binomial, the probability of the code being able to correct exactly  $s$  erasures, but no more, may also be simplified from (4) and (5).

$$\Pr(s) = \sum_{j=1}^{s+1} \frac{n!}{(n-j)!j!2^{n-k}} \frac{(n-j)!(s+1)!}{n!(s+1-j)!} - \sum_{j=1}^s \frac{n!}{(n-j)!j!2^{n-k}} \frac{(n-j)!s!}{n!(s-j)!} \quad (15)$$

which simplifies to

$$\Pr(s) = \frac{2^s - 1}{2^{n-k}} \quad (16)$$

for  $s < n - k$  and for  $s = n - k$

$$\Pr(n - k) = 1 - \sum_{j=1}^{n-k} \frac{n!}{(n-j)!j!2^{n-k}} \frac{(n-j)!(n-k)!}{n!(n-k-j)!} \quad (17)$$

and

$$\Pr(n - k) = \frac{1}{2^{n-k}} \quad (18)$$

For codes whose weight enumerator coefficients are approximately binomial, the pdf of correctable erasures is given in Table 1.

The probability of codeword decoder error for these codes is given by substitution into (9),

$$P_{\text{code}}(p) = \sum_{s=0}^{n-k} \left( \frac{2^s - 1}{2^{n-k}} \right) \times \frac{n!}{(n-s)!s!} p^s (1-p)^{(n-s)} + P_{\text{MDS}}(p) \quad (19)$$

As first shown by Dumer and Farrell [10] when  $n$  tends to  $\infty$ , these codes achieve the erasure channel capacity. As examples, the probability of codeword decoder error for hypothetical rate 0.9 codes, having binomial weight

**Table 1: PDF of number of correctable erasures for codes whose weight enumerator coefficients are binomial**

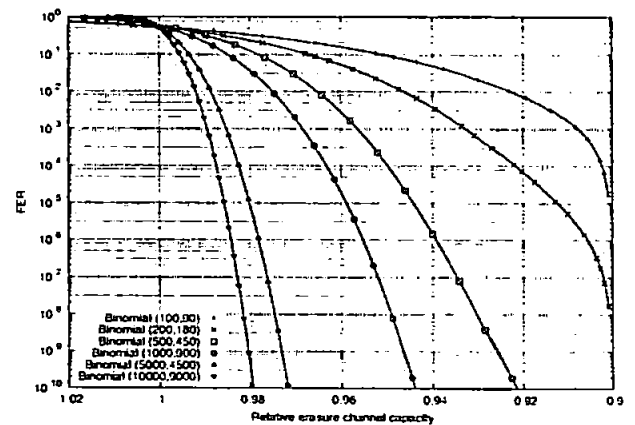
Correctable erasures	Probability
$n - k$	$\frac{1}{2^{n-k}}$
$n - k - 1$	$0.5 - \frac{1}{2^{n-k}}$
$n - k - 2$	$0.25 - \frac{1}{2^{n-k}}$
$n - k - 3$	$0.125 - \frac{1}{2^{n-k}}$
$n - k - 4$	$0.0625 - \frac{1}{2^{n-k}}$
$n - k - 5$	$0.03125 - \frac{1}{2^{n-k}}$
$n - k - 6$	$0.0150625 - \frac{1}{2^{n-k}}$
$n - k - 7$	$0.007503125 - \frac{1}{2^{n-k}}$
$\vdots$	$\vdots$
$n - k - s$	$\frac{1}{2^s} - \frac{1}{2^{n-k}}$

distributions, and lengths 100–10000 bits are shown plotted in Fig. 1 as a function of the channel erasure probability expressed in terms of relative erasure channel capacity  $0.9/(1-p)$ . It can be seen that at a decoder error rate of  $10^{-8}$  the (1000, 900) code is operating at 95% of channel capacity, and the (10000, 9000) code is operating at 98% of channel capacity. A comparison with MDS codes is shown in Fig. 2. For code-lengths from 500 to 50000 bits, it can be seen that for code-lengths of 5000 bits and above, these rate 0.9 codes are optimum since their performance is indistinguishable from the performance of MDS codes with the same length and rate.

#### 4 Results for particular codes

The first example is the extended BCH code (128, 99, 10) whose coefficients up to weight 30 of the weight enumerator polynomial [17] are tabulated in Table 2.

The pdf of the number of erased bits that are correctable up to the maximum of 29 erasures, derived from (1), is shown in Fig. 3a. Also shown in Fig. 3a is the performance obtained numerically. It is straightforward, by computer simulation, to evaluate the erasure correcting performance of the code by generating a pattern of erasures randomly



**Fig. 1 FER performance of codes with binomial weight enumerator coefficients**

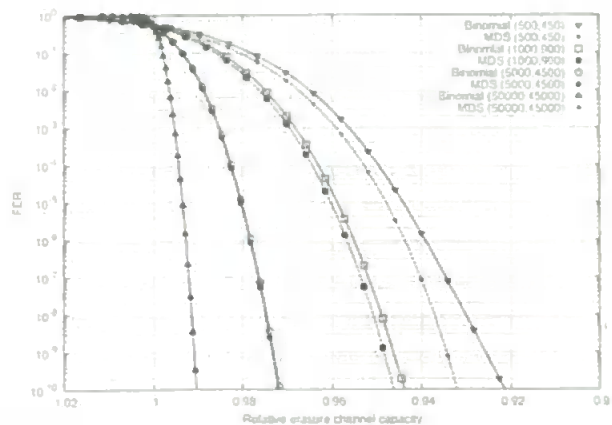


Fig. 2 Comparison of codes with binomial weight enumerator coefficients to MDS codes

and solving these in turn by using the parity check equations. This procedure corresponds to ML decoding [6, 7]. Moreover, the codeword responsible for any instances of non-MDS performance (due to this erasure pattern) can be determined by back substitution into the solved parity check equations. Except for short codes or very high rate codes, it is not possible to complete this procedure exhaustively, because there are too many combinations of erasure patterns. For example, there are  $4.67 \times 10^{28}$  combinations of 29 erasures in this code of length 128 bits. In contrast, there are relatively few low-weight codewords responsible for the non-MDS performance of the code. For example, each codeword of weight 10 is responsible for  $\binom{118}{19} = 4.13 \times 10^{21}$  erasures patterns not being solvable.

As the  $d_{\min}$  of this code is 10, the code is guaranteed to correct any erasure pattern containing up to 9 erasures. It

Table 2: Low-weight spectral terms for the extended BCH (128, 99) code

Weight	$A_d$
0	1
10	796544
12	90180160
14	6463889536
16	347764539928
18	14127559573120
20	445754705469248
22	11149685265467776
24	224811690627712384
26	3704895377802191104
28	50486556173121673600
30	574502176730571255552

can be seen from Fig. 3a that the probability of not being able to correct any pattern of 10 erasures is less than  $10^{-8}$ . The probability of correcting 29 erasures, the maximum number, is 0.29. The average number of erasures corrected is 27.44, almost three times the  $d_{\min}$ , and the average shortfall from MDS performance is 1.56 erased bits. The prediction of performance by the lower bound is pessimistic due to double codeword counting in erasure patterns featuring more than 25 bits. The effect of this is evident in Fig. 3a. The lower bound average number of erasures corrected is 27.07 and the shortfall from MDS performance is 1.93 erasures, an error of 0.37 erasures. The erasure performance evaluation by simulation is complementary to the analysis using the weight distribution of the code, in which the simulation being a sampling procedure is inaccurate for short, uncorrectable erasure patterns, because few codewords are responsible for the

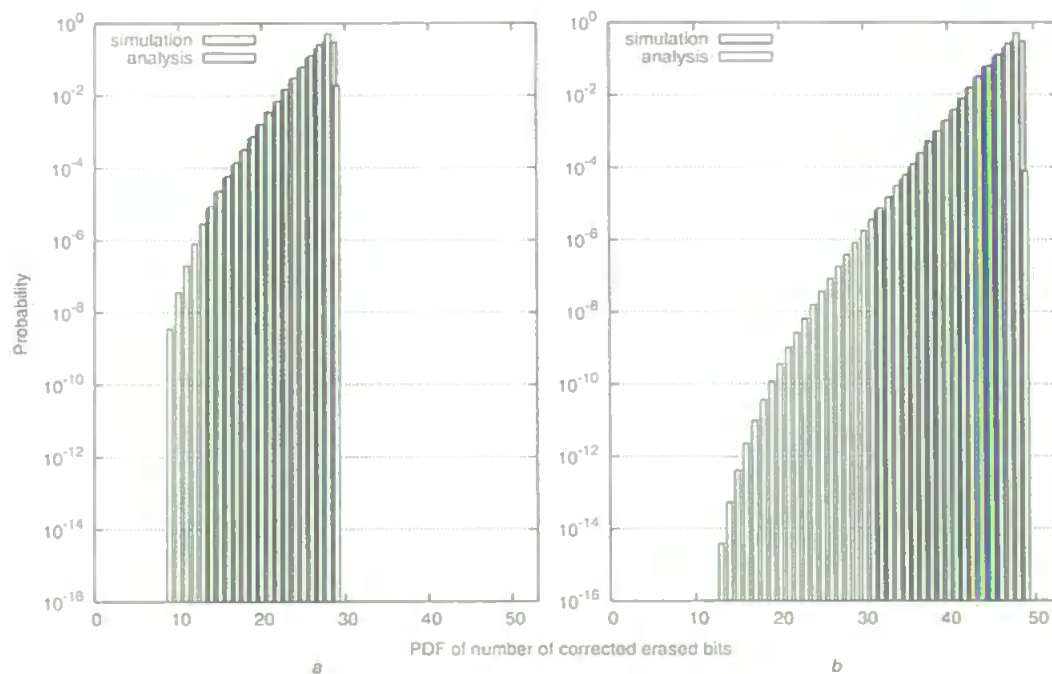


Fig. 3 PDF of erasure corrections

a (128, 99, 10) extended BCH code  
b (256, 207, 14) extended BCH code

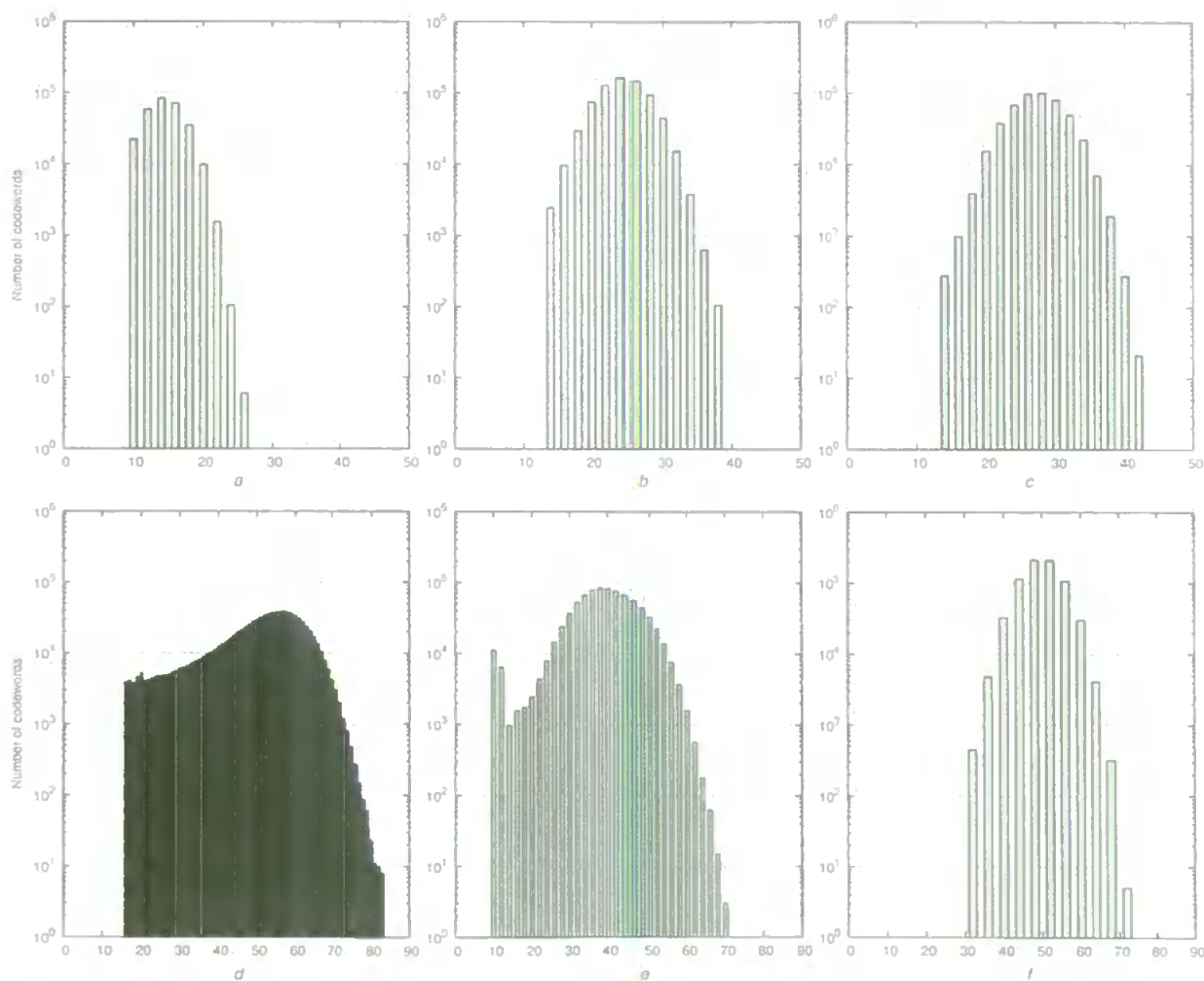
performance in this region. For short, uncorrectable erasure patterns, the lower bound analysis is tight in this region as it is not possible for these erasure patterns to contain more than one codeword because of codewords differing by at least  $d_{\min}$ .

The distribution of the codeword weights responsible for non-MDS performance of this code is shown in Fig. 4a. This is in contrast to the distribution of low-weight codewords shown in Fig. 5. Although there are a larger number of higher weight codewords there is less chance of an erasure pattern containing a higher weight codeword. The maximum occurrence is for weight 14 codewords as shown in Fig. 4a.

The FER performance of the BCH (128, 99, 10) code is shown in Fig. 6 as a function of relative capacity defined by  $k/(1-p)n$ . Also shown in Fig. 6 is the FER performance of a hypothetical (128, 99, 30) MDS code. Equations (8) and (9) were used to derive Fig. 6. As shown in Fig. 6, there is a significant shortfall in capacity even for the optimum MDS code. This shortfall is attributable to the relatively short length of the code. At  $10^{-9}$  FER the BCH (128, 99, 10) code achieves approximately 80% of the erasure channel capacity. The maximum capacity achievable by any (128,

99) binary code as represented by a (128, 99, 30) MDS code is  $\sim 82.5\%$ .

An example of a longer code is the (256, 207, 14) extended BCH code. The coefficients up to weight 50 of the weight enumerator polynomial [18] are tabulated in Table 3. The evaluated erasure correcting performance of this code is shown in Fig. 3b and the code is able to correct up to 49 erasures. It can be seen from Fig. 3b that there is a close match between the lower bound analysis and the simulation results for the number of erasures between 34 and 46. Beyond 46 erasures the lower bound becomes increasingly pessimistic due to double counting of codewords. Below 34 erasures the simulation results are erratic due to insufficient samples. It can be seen from Fig. 3b that the probability of correcting only 14 erasures is  $< 10^{-13}$  (actually  $5.4 \times 10^{-14}$ ) even though the  $d_{\min}$  of the code is 14. If a significant level of erasure correcting failures is defined as  $10^{-6}$  then from Fig. 3b this code is capable of correcting up to 30 erasures even though the guaranteed number of correctable erasures is only 13. The average number of erasures correctable for the code is 47.4 and the average shortfall is 1.6 erased bits. The distribution of codeword weights responsible for the non-MDS performance of this code is shown in Fig. 4b.



**Fig. 4** Distribution of codeword weights responsible for non-MDS performance

- a (128, 99, 10) extended BCH code
- b (256, 207, 14) extended BCH code
- c (512, 457, 14) extended BCH code
- d (240, 120, 16) turbo code
- e (200, 100, 10) PEG LDPC code
- f (200, 100, 32) extended quadratic residue code

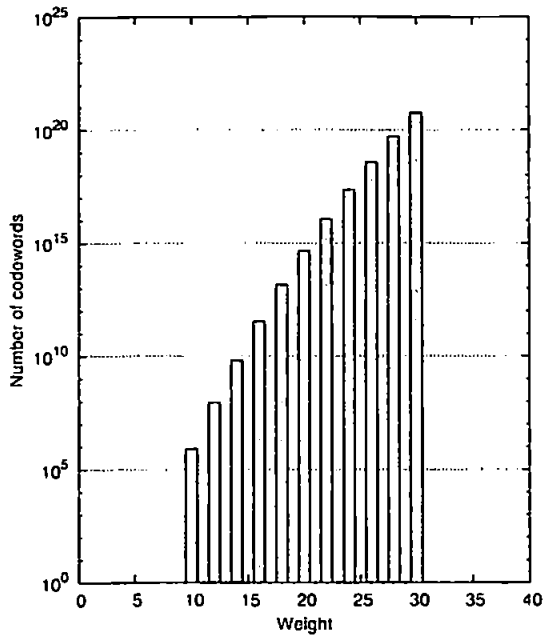


Fig. 5 Distribution of low-weight codewords for the (128, 99, 10) extended BCH code

The FER performance of the BCH (256, 207, 14) code is shown in Fig. 6 as a function of relative capacity defined by  $k/(1-p)n$ . Also plotted in Fig. 6 is the FER performance of a hypothetical (256, 207, 50) MDS code. Equations (8) and (9) were used to derive this. As observed from Fig. 6 there is less shortfall in capacity compared to the BCH (128, 99, 10) code. At  $10^{-9}$  FER, the BCH (256, 207, 14) code achieves approximately  $\sim 85.5\%$  of the erasure channel capacity. The maximum capacity achievable by any (256, 207) binary code as represented by (256, 207, 50) hypothetical MDS code is  $\sim 87\%$ .

The next code to be investigated is the (512, 457, 14) extended BCH code, which was chosen because it is comparable to the (256, 207, 14) code in being able to correct a similar maximum number of erasures (55 cf. 49) and has the same  $d_{\min}$  of 14. Unfortunately, the weight enumerator polynomial is yet to be determined, and only erasure simulation results may be obtained. Fig. 7a shows the performance of this code. The average number of erasures

Table 3: Spectral terms up to weight 50 for the extended BCH (256, 207) code

Weight	$A_d$
0	1
14	159479040
16	36023345712
18	6713050656000
20	996444422768640
22	119599526889384960
24	11813208348266177280
26	973987499253055749120
28	67857073021007558686720
30	4036793565003066065373696
32	206926366333597318696425720
34	9212465086525810564304939520
36	358715843060045310259622139904
38	12292268362368552720093779880960
40	372755158433879986474102933212928
42	10052700091541303286178365979008000
44	242189310556445744774611488568535040
46	5233629101357641331155176578460897024
48	101819140628807204943892435954902207120
50	1789357109760781792970450788764603959040

corrected is 53.4 and the average shortfall is 1.6 erased bits. The average shortfall is identical to the (256, 207, 14) extended BCH code. Also the probability of achieving MDS code performance, that is being able to correct all  $n - k$  erasures is also the same and equal to 0.29. The distribution of codeword weights responsible for non-MDS performance of the (512, 457, 14) code is very similar to the (256, 207, 14) code as shown in Fig. 4c.

An example of an extended cyclic quadratic residue code is the (168, 84, 24) code whose coefficients of the weight enumerator polynomial are given in [19] and are tabulated up to weight 72 in Table 4. This code is a self-dual, doubly even code, but not extremal because its  $d_{\min}$  is not 32 but 24 [20]. The FER performance of the (168, 84, 24) code is shown in Fig. 6 as a function of relative capacity defined by  $k/(1-p)n$ . Also plotted in Fig. 6 is the FER performance of a hypothetical (168, 84, 85) MDS code. Equations (8) and (9) were used to derive this. The performance of the (168, 84, 24) code is close to that of the hypothetical MDS code but both codes are around 30% from capacity at  $10^{-6}$  FER.

The erasure correcting performance of non-algebraic designed codes is quite different from algebraic designed codes as can be seen from the performance results of a (240, 120, 16) turbo code shown in Fig. 7b. The turbo code features memory four constituent recursive encoders and a code matched, modified  $\mathcal{S}$  interleaver, to maximise the  $d_{\min}$  of the code. The average number of erasures correctable by the code is 116.5 and the average shortfall is 3.5 erased bits. The distribution of codeword weights responsible for non-MDS performance of the (240, 120, 16) code is very different from the algebraic codes and features a flat distribution as shown in Fig. 4d.

Similarly, the erasure correcting performance of a (200, 100, 10) LDPC code designed using the PEG algorithm [14] is again quite different from the algebraic codes as shown in Fig. 7c. As is typical of randomly generated LDPC codes, the  $d_{\min}$  of the code is quite small at 10,

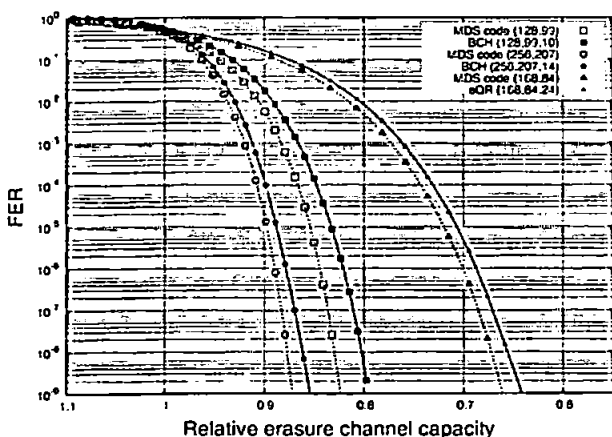
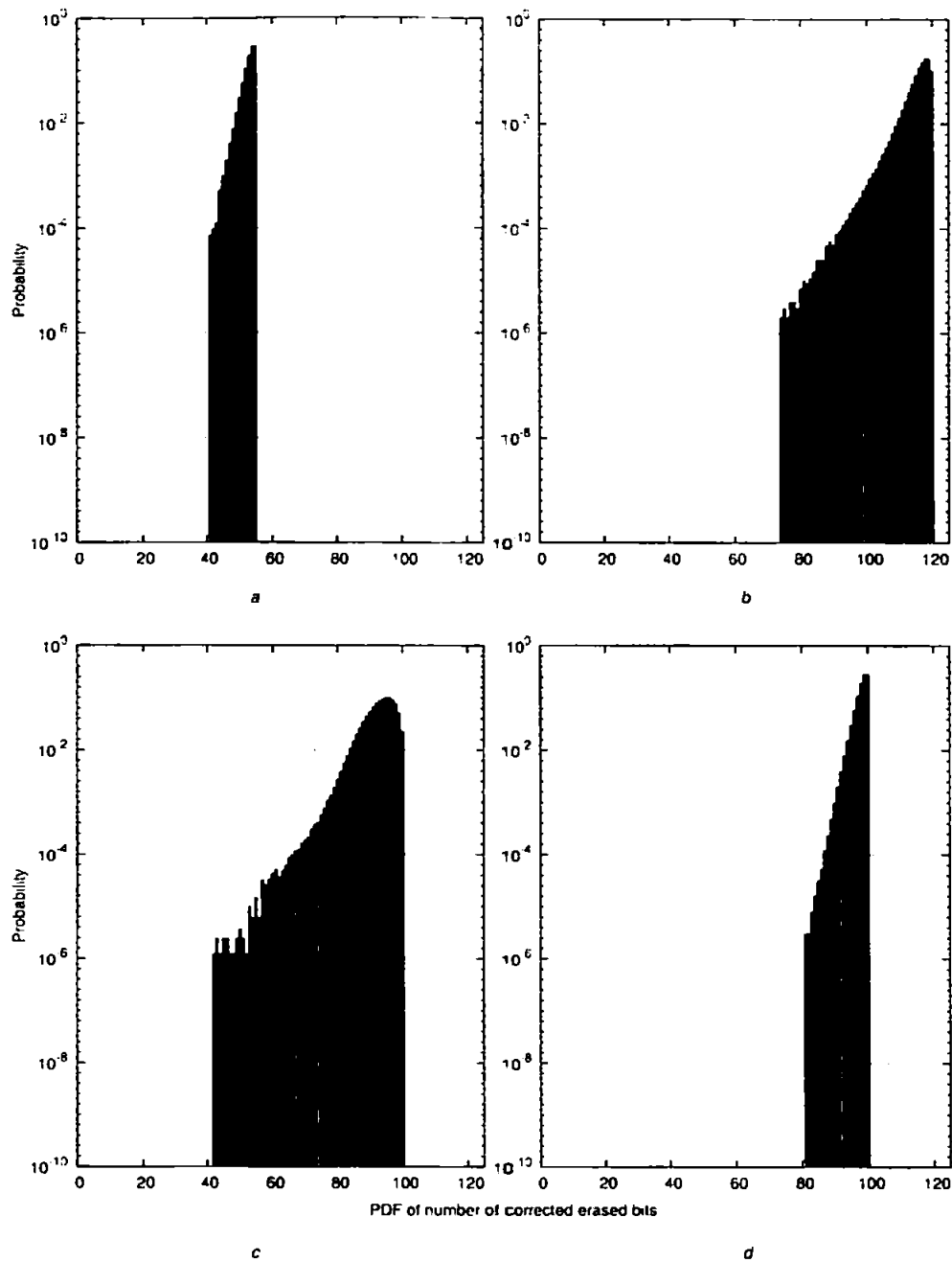


Fig. 6 FER performance for the (128, 99, 10) and (256, 207, 14) extended BCH codes and (168, 84, 24) extended quadratic residue code for the erasure channel



**Fig. 7** PDF of erasure corrections

- a* (512, 457, 14) extended BCH code
- b* (240, 120, 16) turbo code
- c* (200, 100, 10) PEG LDPC code
- d* (200, 100, 32) extended quadratic residue code

even though the code has been optimised. For this code, the average number of correctable erasures is 93.19 and the average shortfall is 6.81 erased bits. This is markedly worse than the turbo code performance. It is the preponderance of low weight codewords that is responsible for the inferior performance of this code compared to the other codes as shown by the codeword weight distribution in Fig. 4*e*.

The relative weakness of the LDPC code and turbo code becomes clear when compared to a good algebraic code with similar parameters. There is a (200, 100, 32) extended quadratic residue code. The pdf of the number of erasures corrected by this code is shown in Fig. 7*d*. The difference between having a  $d_{\min}$  of 32 compared to 16 for the turbo code and 10 for the LDPC code is dramatic. The average

number of correctable erasures is 98.4 and the average shortfall is 1.6 erased bits. The weight enumerator polynomial of this self-dual code, is currently unknown as evaluation of the  $2^{100}$  codewords is currently beyond the reach of today's computers. However, the distribution of codeword weights responsible for non-MDS performance of the (200, 100, 32) code which is shown in Fig. 4*f* indicates the doubly even codewords of this code and the  $d_{\min}$  of 32.

## 5 Determination of the $d_{\min}$ of a linear code

It is well known that the determination of weights of any linear code is a non-deterministic polynomial time hard

**Table 4: Spectral terms up to weight 72 for the extended quadratic residue (168, 84) code**

Weight	$A_d$
0	1
24	776216
28	18130188
32	5550332508
36	1251282702264
40	166071600559137
44	13047136918828740
48	629048543890724216
52	19087130695796615088
56	372099690249351071112
60	4739291519495550245228
64	39973673337590380474086
68	225696677727188690570184
72	860241108921860741947676

problem [21] and except for short codes, the best methods for determining the minimum Hamming distance,  $d_{\min}$  codeword of a linear code, to date, are probabilistically based [22]. Most methods are based on the generator matrix, the  $G$  matrix of the code and tend to be biased towards searching, using constrained information weight codewords. Such methods become less effective for long codes or codes with code rates around 1/2 because the weights of the evaluated codewords tend to be binomially distributed with average weight  $n/2$  [9].

Corollary 2 from Section 2, provides the basis of a probabilistic method to find low-weight codewords in a significantly smaller search space than the  $G$  matrix methods. Given an uncorrectable erasure pattern of  $n - k$  erasures, from Corollary 2, the codeword weight is less than or equal to  $n - k$ . The search method suggested by this, becomes one of randomly generating erasure patterns of  $n - k + 1$  erasures, which of course are uncorrectable by any  $(n, k)$  code, and determines the codeword and its weight from (2). This time, the weights of the evaluated codewords will tend to be binomially distributed with average weight  $n - k + 1/2$ . With this trend, for  $N_{\text{trials}}$  the number of codewords determined with weight  $d$ ,  $M_d$  is given by

$$M_d = N_{\text{trials}} \frac{(n - k + 1)!}{d!(n - k - d + 1)!2^{n-k+1}} \quad (20)$$

As an example of this approach, the self-dual, bordered, double circulant code (168, 84) based on the prime number 83 is considered. This code was described in [23] as having an unconfirmed  $d_{\min}$  of 28. From (20) when using 18000 trials, 10 codewords of weight 28 will be found on average. However, as the code is doubly even and only has codewords weights which are a multiple of 4, using 18000 trials, 40 codewords are expected. In a set of trials using this method for the (168,84) code, 61 codewords of weight 28 were found with 18000 trials. Furthermore, 87 codewords of weight 24 were also found indicating that the  $d_{\min}$  of this code is 24.

The search method can be improved by biasing towards the evaluation of erasure patterns that have small numbers of erasures that cannot be solved. Recalling the analysis in Section 2, as the parity check equations are Gaussian reduced, no erased bit is a function of any other erased

bits. There will be  $n - k - s$  remaining parity check equations, which do not contain the erased bit co-ordinates  $x_j$ . The remaining equations may be searched to see if there is an unerased bit co-ordinate that is not present in any of the equations. If there is one such co-ordinate, then this co-ordinate in conjunction with the erased co-ordinates solved so far forms an uncorrectable erasure pattern involving only  $s$  erasures instead of  $n - k + 1$  erasures. With this procedure, biased towards small numbers of unsolvable erasures, it was found that for the above code, 21 distinct codewords of weight 24 and 17 distinct codewords of weight 28 were determined in 1000 trials and the search took 8 s on a typical 1.6 GHz personal computer (PC).

In another example the (216, 108) self-dual, bordered double circulant code is given in [23] with an unconfirmed  $d_{\min}$  of 36. With 1000 trials which took 26 s on the PC, 11 distinct codewords were found with weight 24 and a longer evaluation confirmed that the  $d_{\min}$  of this code is indeed 24.

## 6 Conclusions

Analysis of the erasure correcting performance of linear, binary codes has provided the surprising result that many codes can correct, on average, almost  $n - k$  erasures and have a performance close to the optimum performance as represented by (hypothetical) binary MDS codes.

It was shown for codes having a weight distribution approximating to a binomial distribution, and this includes many common codes, such as BCH codes, Goppa codes and self-dual codes, that these codes can correct at least  $n - k - 2$  erasures on average and closely match the FER performance of MDS codes as code-lengths increase. The asymptotic performance achieves capacity for the erasure channel. It was also shown that codes designed for iterative decoders, the turbo and LDPC codes, are relatively weak codes for the erasure channel and compare poorly with algebraically designed codes. Turbo codes, designed for optimised  $d_{\min}$  were found to outperform LDPC codes.

Determination of the erasure correcting performance of a code provides a means of determining the  $d_{\min}$  of a code and an efficient search method was described. Using the method, the  $d_{\min}$  results for two self-dual codes, whose  $d_{\min}$  values were previously unknown were determined, and these codes were found to be (168, 84, 24) and (216, 108, 24) codes.

## 7 Acknowledgments

This work was partly funded by an Overseas Research Students Award Scheme (ORSAS). The authors would like to thank the reviewers for their comments, which have enhanced the quality of presentation of this paper. The rapid and efficient processing of this paper by the Editorial Office, in particular Prof. Habib Rashvand, is also much appreciated.

## 8 References

- 1 Rizzo, L.: 'Effective erasure codes for reliable computer communication protocols'. *ACM SIGCOMM Comput. Commun. Rev.*, 1997, 27, (2), pp. 24–36
- 2 Cai, J., Tjhai, C., Tomlinson, M., Ambroze, M., and Ahmed, M.: 'An efficient solution to packet loss: erasure correcting codes' Palau Salvador, C.E. Proc. 4th IASTED Int. Conf. on Communication Systems and Networks (ACTA Press, 2005), pp. 224–229
- 3 Cai, J., Tomlinson, M., Tjhai, C., Ambroze, M., and Ahmed, M.: 'Advances in iterative decoding and maximum likelihood decoding for the packet network with comparisons to fountain codes over the erasure channel'. Proc. 4th Int. Symp. on Turbo Codes in connection with 6th Int. ITG-Conf. on Source and Channel Coding, Munich, Germany, April 2006, pp. 3–7

- 4 Cai, J., Tomlinson, M., Tjhai, C., Ambroze, M., and Ahmed, M.: 'Comparison of concatenated Reed-Solomon coding with hard-decision to soft-decision LDPC coding for the Rayleigh fading channel'. Proc. IEEE Inf. Theory Workshop, Chengdu, China, 22–26 October 2006, pp. 135–139
- 5 Rabin, M.O.: 'Efficient dispersal of information for security, load balancing, and fault tolerance', *J. ACM*, 1989, 36, (2), pp. 335–348
- 6 Pishro-Nik, H., and Fekri, F.: 'On decoding of low-density parity-check codes over the binary erasure channel', *IEEE Trans. Inf. Theory*, 2004, 50, (3), pp. 439–454
- 7 Di, C., Proietti, D., Telatar, I., Richardson, T., and Urbanke, R.: 'Finite-length analysis of low-density parity-check codes on the binary erasure channel', *IEEE Trans. Inf. Theory*, 2002, 48, (6), pp. 1570–1579
- 8 Peterson, W., and Weldon, Jr., E.J.: 'Error-correcting codes' (MIT Press, 1972)
- 9 MacWilliams, F.J., and Sloane, N.J.A.: 'The theory of error-correcting codes' (North-Holland, 1977), ISBN 0 444 85193 3
- 10 Dumer, I., and Farrell, P.: 'Erasure correction performance of linear block codes' Cohen, G., Litsyn, S., Lobstein, A., and Zemor, G. (Eds.): 'Lecture Notes in Computer Science' (Springer-Verlag, 1993), vol. 781, pp. 316–326
- 11 Krasikov, I., and Litsyn, S.: 'On spectra of BCH codes', *IEEE Trans. Inf. Theory*, 1995, 41, (3), pp. 786–788
- 12 Krasikov, I., and Litsyn, S.: 'On the accuracy of the binomial approximation to the distance distribution of codes', *IEEE Trans. Inf. Theory*, 1995, 41, (5), pp. 1472–1474
- 13 Roychowdhury, V., and Vatan, F.: 'Bounds for the weight distribution of weakly self-dual codes', *IEEE Trans. Inf. Theory*, 2001, 47, (1), pp. 393–396
- 14 Hu, X.Y., Eleftheriou, E., and Arnold, D.M.: 'Regular and irregular progressive edge-growth tanner graphs', *IEEE Trans. Inf. Theory*, 2005, 51, (1), pp. 386–398
- 15 Douillard, C., and Berrou, C.: 'Turbo codes with rate- $m/(m+1)$  constituent convolutional codes', *IEEE Trans. Commun.*, 2005, 53, (10), pp. 1630–1638
- 16 Elias, P.: 'Coding for two noisy channels' in 'Third London symposium on information theory' (Academic Press, New York, 1956)
- 17 Desaki, Y., Fujiwara, T., and Kasami, T.: 'The weight distribution of extended binary primitive BCH code of length 128', *IEEE Trans. Inf. Theory*, 1997, 43, (4), pp. 1364–1371
- 18 Fujiwara, T., and Kasami, T.: 'The weight distribution of (256,  $k$ ) extended binary primitive BCH code with  $k \leq 63$ ,  $k \geq 207$ ', Technical Report of IEICE, 1993, IT97-46, pp. 29–33
- 19 Tjhai, C., Tomlinson, M., Horan, R., Ahmed, M., and Ambroze, M.: 'On the efficient codewords counting algorithm and the weight distribution of the binary quadratic double-circulant codes', Proc. IEEE Inf. Theory Workshop, Chengdu, China, 22–26 October 2006, pp. 42–46
- 20 Conway, J.H., and Sloane, N.J.A.: 'A new upper bound on the minimum distance of self-dual codes', *IEEE Trans. Inf. Theory*, 1990, 36, (6), pp. 1319–1333
- 21 Dumer, I., Micciancio, D., and Sudan, M.: 'Hardness of approximating the minimum distance of a linear code', *IEEE Trans. Inf. Theory*, 2003, 49, (1), pp. 22–37
- 22 Canteaut, A., and Chabaud, F.: 'A new algorithm for finding minimum weight words in a linear code: application to McEliece's cryptosystem and to narrow-sense BCH codes of length 511', *IEEE Trans. Inf. Theory*, 1998, 44, (1), pp. 367–378
- 23 Gulliver, T.A., and Senkevitch, N.: 'On a class of self-dual codes derived from quadratic residue', *IEEE Trans. Inf. Theory*, 1999, 45, (2), pp. 701–702



# Comparison of Concatenated Reed-Solomon Coding with Hard-decision to Soft-decision LDPC Coding for the Rayleigh Fading Channel

J. Cai, M. Tomlinson, C. Tjhai, M. Ambroze, and M. Ahmed  
Fixed and Mobile Communications Research, University of Plymouth,  
PL4 SAA, United Kingdom  
{jcai,mtomlinson,ctjhai,mambroze,mahmed}@plymouth.ac.uk

*Abstract* — Reed-Solomon (RS) codes  $C(n, k, n - k + 1)$  are commonly used error-control codes, because they are Maximum Distance Separable (MDS) codes. Over a Binary Erasure Channel, RS codes perform with optimal results and approach the maximum channel capacity. In this paper, we apply RS codes to packet wireless transmission over the uncorrelated flat Rayleigh fading channel. It is shown that by using an RS code concatenated with BCH codes and using hard decisions, better results are obtained than using bit interleaved LDPC codes, with soft-decision decoding. The BCH code is used to correct small numbers of errors due to noise and also to detect the presence of deep fades, in which case the entire packet is erased. Erased packets are corrected by the RS code. We also discuss the effect of overall code rate on the net performance.

## I. INTRODUCTION

The Rayleigh fading Channel is widely used as a model of wireless communications. Channel coding techniques are a powerful tool to improve the reliability and efficiency of wireless communications. From [7], [8] and [9], the performance on Rayleigh fading channels and turbo codes design for Rayleigh fading channels have been given and explored. In [10], irregular low density parity-check (LDPC) codes have also been applied to an uncorrelated flat Rayleigh fading Channel, and shown to outperform turbo codes over a wide range of mobile speeds. Reed Solomon (RS) codes have also been used for the Rayleigh fading channel, as the fading channel causes error bursts.

RS codes [1] are classical, commonly used error-control codes with a wide range of applications in modern communications. They constitute an efficient class of linear codes using multi-bit symbols and have the capability of correcting/detecting symbol errors and correcting symbol erasures. It is well known that an RS code  $C(n, k, d)$ , where  $n$  is the code length,  $k$  is the information length and  $d$  is the Hamming distance of  $C$ , can correct up to  $t = \lfloor (n - k)/2 \rfloor$  random symbol errors, and correct up to  $n - k$  symbol erasures. The classical algorithms of Berlekamp [4] and Massey [13] can correct  $t$  errors and  $\epsilon$  erasures when  $2t + \epsilon \leq n - k$ , which can achieve the error bound  $p = \frac{n-k+1}{2}$  with running time  $O(n^2)$ . In [5], it was presented that a polynomial time list decoding algorithm for Reed-Solomon codes can correct more

than  $(n - k)/2$  errors, provided  $k < n/3$ . Using the Roth and Ruckenstein [6] algorithm, the same bound can be achieved with running time  $O(n^2 \log^2 n)$ . Since more erasures than errors can be corrected, it is advantageous to determine the reliability of the received RS-coded symbols and to erase the low-reliability symbols prior to the decoding process. In this paper, we apply the concatenation of RS and BCH codes to packet wireless transmission over the uncorrelated flat Rayleigh fading channel. BCH codes of different code-rates and minimum Hamming distance are used to correct small numbers of bit errors in the packet transmission. Also the BCH code is used to detect relatively deep fades by error detecting relatively large numbers of errors in the packet. In this case the entire packet is erased. Erased packets are corrected by the RS outer code. To obtain best results for the Rayleigh fading channel, we use the In-place decoder [2] for erasure correction by the RS code. The paper is organised as follows. Section II briefly reviews the system and channel model. In Section III, we describe the product packetisation method. In Section IV, we give an analysis of a concatenated RS code over the Rayleigh fading channel. In Section V, we describe the implementation by using concatenated RS codes and analyse the computational complexity on both encoding and decoding. In Section VI, numerical Frame Error Rate(FER) results are given for these codes in comparison to soft-decision decoding of LDPC codes as a function of the overall concatenated code rate. The conclusions are given in Section VII.

## II. SYSTEM AND CHANNEL MODEL

The transmitted sequence  $x_k$  passes through a discrete time frequency non-selective Rayleigh fading channel with additive white noise, where  $k$  is an integer symbol index,  $x_k$  is a binary-phased-shift-keying (BPSK) symbol with amplitude  $\pm\sqrt{E_x}$ . The received discrete-time baseband signal can be written as  $y_k = \alpha_k x_k + n_k$ , where  $\alpha_k$  is the Rayleigh distributed fading coefficient with  $E(\alpha_k^2) = 1$  and  $n_k$  is a complex white noise sample with variance  $N_0/2$  per dimension.

The probability density function (pdf) of the output  $y$  can be described as:

$$p(y|\omega, \alpha) = \frac{1}{\sqrt{2\pi\delta^2}} \exp\left(-\frac{(y - \omega \cdot \alpha)^2}{2\delta^2}\right). \quad (1)$$

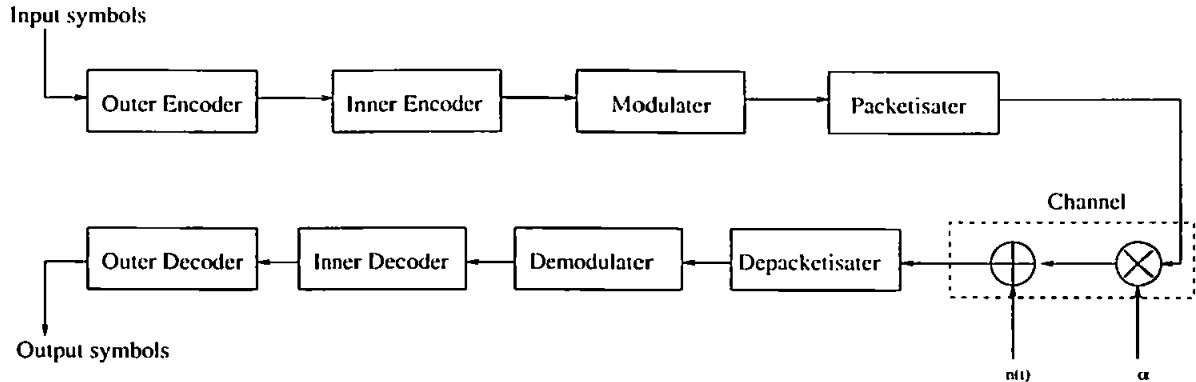


Figure 1: Transmission System Diagram

where  $\omega = (1 - 2x)$  is the binary input after the BPSK modulation,  $\delta^2 = (\frac{1}{2R} \cdot (E_b/N_0))$ , and  $R$  is the code rate.

### III. PRODUCT PACKETISATION AND PROTOCOL DESCRIPTION

In this paper, we use the product packetisation method to arrange symbols into packets. Instead of placing the symbols into the sequential packets, we first split the data into blocks corresponding to codewords of the RS erasure correcting code, encode them and then packetise symbols from the same positions of each codeword. Denoting the length of input as  $l_d$  and the payload size of each packet as  $l_p$ , the minimum number of packets needed can be calculated by  $l_d/l_p$ . Hereafter, a RS block code  $C$  with this scheme is used with the number of information bits  $k = l_p$  and a code rate  $R$ . For a linear block code  $C(n, k)$ , there are  $k$  information packets and  $n - k$  parity-check packets, but the benefit of using the RS codes is that only  $k$  correct packets need to be received and these can be any  $k$  packets. For convenience, the first  $k$  symbols are termed information symbols. The  $i$ -th packet contains  $l_d$  information symbols whose positions are at  $j \cdot k + i$ , where  $j = 0, 1, \dots, (l_d - 1)$  and  $0 \leq i < k$ . At the receiver, all the received information packets need to be restructured into a buffer  $\mathbf{X}$  with length  $l_d$ . If the  $i$ -th packet is received, the symbols contained in the packet should be placed in the buffer at the positions of  $x_{j \cdot k + i}$ , where  $j = 0, \dots, k - 1$ ; if it has been erased during the transmission, '?'s will be placed in the positions of  $x_{j \cdot k + i}$ , where  $j = 0, \dots, k - 1$  to mark the symbols as erased.

The packets are transmitted continuously and at the receiver, they are depacketisated and decoded into a buffer with a length of  $l_d$ .

### IV. ANALYSIS OF CONCATENATED RS CODES

The most popular decoding algorithm of RS codes is called "error-and-erasure" decoding algorithm, which is preferable to "error-correction-only" decoding algorithm. With this algorithm, an RS code is capable of correcting

$t$  errors and recovering  $\epsilon$  erasures, under the condition of  $2t + \epsilon \leq n - k$ . Then, we can obtain the probability of decoder failure as follows:

$$P_f = \sum_{\epsilon=0}^{n-k} \left\{ \sum_{t=\lfloor (n-k-\epsilon/2)+1 \rfloor}^{n-\epsilon} P(t|\epsilon)P(\epsilon) \right\} + \sum_{\epsilon=n-k+1}^n P(\epsilon) \quad (2)$$

where  $P(\epsilon)$  is the probability of  $\epsilon$  erasures and  $P(t|\epsilon)$  is the conditional probability of  $t$  errors given  $\epsilon$  erasures in the remaining  $n - \epsilon$  positions, which are defined as follows:

$$P(\epsilon) = \binom{n}{\epsilon} \cdot p_{er}^{\epsilon} (1 - p_{er})^{n-\epsilon}. \quad (3)$$

and

$$P(t|\epsilon) = \binom{n-\epsilon}{t} \cdot p_e^t (1 - p_e)^{n-\epsilon-t} \quad (4)$$

where  $p_{er}$  is the probability of an erasure, and  $p_e$  is the probability of an error but not an erasure.

To ensure that the system can give good performance in the Rayleigh fading channel, the BCH code is used to correct and to detect multiple hard decision errors in each received packet. If the number of errors is small, they are corrected by the Hamming or BCH code. If the number of errors exceeds a threshold, the entire packet is erased and corrected by the RS code. It is well known that a RS code can recover  $n - k$  erasures if a Maximum Likelihood decoding algorithm is implemented. In [2], we introduced a complexity-reduced optimal decoding algorithm - the In-place Algorithm. It is always able to solve the maximum number of erasures correctable by the code. However, for a Rayleigh fading channel, we also need to consider the Gaussian noise and fading factor which decrease the energy of each symbol. The system is shown in Figure 1.

## V. EFFICIENT RS CODES ENCODER/DECODER IMPLEMENTATION

### A. Encoder

In this section, we describe a way to encode the input symbols in an "encoding-on-the-fly" manner. Linear block codes can be generated from their parity-check matrix, denoted by  $H$ , which is usually, but not always, in the form of a row-echelon reduced  $H$  matrix. Each row of the  $H$  matrix can be associated with a parity-check equation. For convenience, we denote this parity check equation by  $\mathbf{h}_i$  where the subscript  $i$  indicates the  $i$ th row of the  $H$  matrix.

Each parity-check equation  $\mathbf{h}_i$  can generate a parity-check  $p_i$  independently for a row-echelon reduced  $H$  matrix. For an  $(n, k)$  RS code  $C$  over  $GF(q)$ , and  $n = q - 1$  and  $q$  is a prime number or a power of a prime number. The parity check matrix of an RS code can be represented by:

$$H = \begin{bmatrix} 1 & \alpha & \alpha^2 & \dots & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \dots & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \alpha^{d_{\min}-1} & \alpha^{2(d_{\min}-1)} & \dots & \alpha^{(n-1)(d_{\min}-1)} \end{bmatrix} \quad (5)$$

where  $\alpha$  is a generator of  $GF(q)$ .

This  $H$  matrix is not efficient for our purpose because every parity check symbol requires a calculation using  $n$  symbols. From Theorem 11-9 in [13], there exist cyclic MDS codes over  $GF(q)$ . In order to efficiently perform encoding on-the-fly, we will use the cyclic form of the  $H$  matrix. For this, the parity check polynomial  $h(x)$  of the RS code is used.

Define the set of powers of consecutive roots  $\alpha$  as  $\Gamma = \{1, 2, \dots, d_{\min} - 1\}$ . The parity-check polynomial  $h(x)$  is given by

$$h(x) = \prod_{i=0, i \notin \Gamma}^{n-1} (x - \alpha^i) = \sum_{i=0}^k \beta_i x^i. \quad (6)$$

from which we can obtain the  $H$  matrix

$$H = \begin{bmatrix} \beta_0 & \beta_1 & \dots & \beta_k & 0 & \dots & \dots \\ 0 & \beta_0 & \beta_1 & \dots & \beta_k & 0 & \dots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & \beta_0 & \beta_1 & \dots & \beta_k \end{bmatrix}. \quad (7)$$

### B. Decoder

Let  $\mathbf{x}'$  denote the received vector. According to [15], optimal decoding is equivalent to solving the linear system, shown in (8). In our case, on average  $k + \epsilon$  packets are transmitted before  $k$  uncrased packets by each destination host.

Accordingly, the following set of equations need to be solved from the  $H$  matrix of the form given in (7).

$$\sum_{i=0}^{k+\epsilon-1} h_{ij} x'_i = 0, \quad j = 0, \dots, \epsilon - 1. \quad (8)$$

As an example, for an RS code, we have (9).

$$\begin{bmatrix} \beta_0 & \beta_1 & \dots & \dots \\ 0 & \beta_0 & \beta_1 & \dots \\ \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \beta_0 & \dots \end{bmatrix} \begin{bmatrix} x'_0 \\ x'_1 \\ \vdots \\ x'_{k+\epsilon-1} \end{bmatrix} = \mathbf{0}. \quad (9)$$

for  $\epsilon$  erasures  $0 \leq \epsilon \leq n - k$ . This linear system can be used for solve for at most  $n - k$  erasures in the case of RS codes. If the equation (9) has a unique solution, an optimal algorithm is possible. Gaussian Reduction algorithm is considered as an optimal algorithm over the BEC, but has a complexity of  $O(N^3)$ . We propose a reduced complexity Gaussian Reduction algorithm - the In-place Algorithm [2] by eliminating the column-permutations required by standard Gaussian Reduction.

Denoting the set of erasure positions as  $\zeta = \{\zeta_0, \zeta_1, \dots, \zeta_\epsilon\}$ , the decoding proceeds in parallel with receiving the packets. (i.e. it is not necessary to wait for all the packets to be received). The  $H$  matrix of the decoder is modified after each packet is received as described below.

We denote the adapted  $H$  matrix at the  $i$ -th erasure detected as  $H'_i$ . When the  $y$ -th erasure  $\zeta_y$  is detected, we flag the first equation which contains  $\zeta_y$  as a flagged equation  $\mathbf{h}_{\zeta_y}$  and leave other equations as unflagged equations. Therefore, at each step in the algorithm, the  $H'$  matrix is divided into two parts:  $H'_x$  which contains all the unflagged equations  $\mathbf{h}_x$  and  $H'_\zeta$  which contains all the flagged equations  $\mathbf{h}_\zeta$ . Then,

$$H' = \{H'_x, H'_\zeta\} \quad (10)$$

And the flagged equation can be denoted as  $\mathbf{h}_{\zeta_y}$ : if  $(\zeta_y \in \mathbf{h}_y) \cap (\mathbf{h}_y \in H'_x) = 1$ ,  $\mathbf{h}_y - \mathbf{h}_{\zeta_y}$ . Then for each  $\mathbf{h}_i$ , where  $(\mathbf{h}_i \in H'_x) \cap (\zeta_y \in (\mathbf{h}_i \cap \mathbf{h}_{\zeta_y})) = 1$ ,

$$h'_{i,j} = h_{\zeta_y,j} - h_{i,j}, \quad j = 0, 1, \dots, n - k - 1. \quad (11)$$

After  $k$  packets are received, the erased symbols are solved. With  $\epsilon$  erasures, for all  $y = 0, \dots, \epsilon - 1$ , if  $\zeta \cap \mathbf{h}_{\zeta_y} = \zeta_y$

$$x_{\zeta_y} = \sum_{j=0}^{n-\epsilon-1} h_{\zeta_y,j} x_j. \quad (12)$$

Then remove  $\mathbf{h}_{\zeta_y}$  from the  $H'_\zeta$  and add it to the  $H'_x$ . Repeat the procedure above until all the erasures are determined.

## VI. NUMERICAL RESULTS

In this section, we compare RS codes, with different code rates, concatenated with different BCH codes with optimal LDPC codes designed using the Progressive Edge

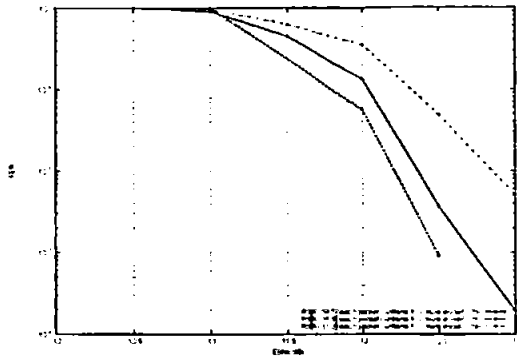


Figure 2: RS codes with variable code rates in the Rayleigh fading channel

Growth (PEG) technique [14] and bit interleaved for transmission over the Rayleigh fading channel.

First, we evaluated the performance of RS codes with different code rates. As shown in Figure 2, the RS(63, 55, 9) code achieved the best performance, as its performance is half order of magnitude better than that of RS(63, 59, 5) code and more than one and a half orders of magnitude better than that of the RS(63, 61, 3) code at a  $FER$  of  $10^{-3}$ .

The simulation results obtained for the RS(63, 59, 5) code concatenated with different BCH codes are given in Figure 3. The BCH codes used were the Hamming (63, 57, 3) code, and the (63, 51, 5), (63, 45, 7) and (63, 18, 10) codes respectively. These codes can detect up to a maximum of 2, 4, 6 and 9 errors, respectively or correct up to a maximum of 1, 2, 3 and 4 errors, respectively or a combination of less corrected/detected errors in each received packet. The performance is improved by using more powerful codes until the rate loss causes significant degradation to the  $E_b/N_0$ . The rate loss for the Hamming (63, 57, 3) code is  $-0.43$  dB; for the (63, 51, 5) code, the tradeoff is  $-0.92$  dB; for the (63, 45, 7) code, the tradeoff is  $-1.46$  dB and for the BCH (63, 18, 10) code, the tradeoff is  $-5.44$  dB. Observe that the concatenated codes have different coding gains over the original RS code, especially the one concatenated with the BCH(63, 45, 7), which has a coding gain of approximately 2.2 dB at a  $FER$  of  $10^{-3}$ . Interestingly, the loss is so excessive that when the (63, 18, 10) code is used, the performance is worse than the uncoded performance. Therefore, the rate loss of the inner codes cannot be ignored.

We also compared the performance between the RS code, concatenated with a BCH code using hard decisions, and the PEG designed LDPC code with soft decision, iterative decoding for the Rayleigh fading channel. In this comparison, we applied the RS codes and the LDPC codes with the same code rate and the same packet size. As shown in Figure 4, the RS with BCH hard-decision decoder achieved a significant performance improvement over the PEG LDPC

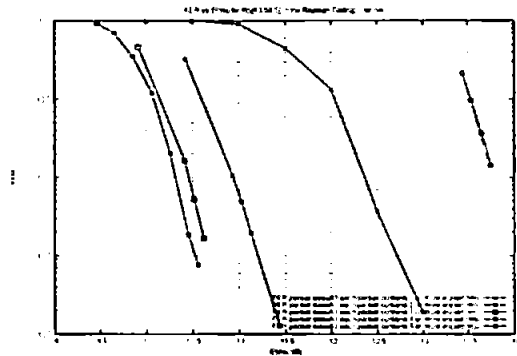


Figure 3: RS (63, 59, 5) concatenated by different Hamming or BCH codes in the Rayleigh fading channel

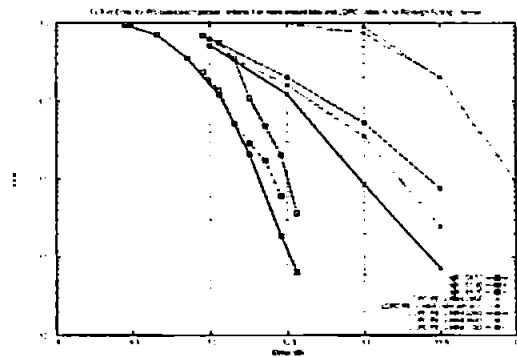


Figure 4: RS with BCH hard-decision codes vs. LDPC soft-decision codes in the Rayleigh fading channel

soft-decision decoder. At a  $FER$  of  $10^{-3}$ , the RS with BCH hard-decision codes can obtain an average coding gain of 2.0 dB over the LDPC soft-decision code in the Rayleigh fading channel. This is attributable to the RS codes being MDS, optimum codes with maximum likelihood erasure correcting decoding. This factor more than compensates for the loss associated with hard decisions for the Rayleigh fading channel particularly as the LDPC codes are not the most powerful codes due to the necessity for iterative decoding.

## VII. CONCLUSION

In this paper, we described the use of RS codes concatenated with BCH codes, with hard-decision decoding for the wireless Rayleigh fading channel. It was shown that the best performance is a function of overall code rate. Furthermore it was shown that the concatenated code combined with simple, hard decision decoding achieves better results than using an optimally designed (PEG) LDPC code combined with soft decision decoding. Further work will provide analysis of the two coding arrangements to show why this is the case. Additionally it will be determined how far the hard decision concatenated arrangement is from capacity for the Rayleigh fading channel.

## REFERENCES

- [1] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Industrial and Applied Mathematics*, vol. 8, pp. 300 - 304, 1960.
- [2] M. Tomlinson, J. Cai, C. Tjhai, M. Ambroze and M. Z. Ahmed, "System for correcting deleted or missing symbols," U. K. Patent application number 0428042.6, Dec. 2004.
- [3] J. Cai, C. Tjhai, M. Tomlinson, M. Ambroze and M. Z. Ahmed, "An Efficient Solution to Packet Loss : Erasure Correcting Codes," in *The Fourth IASTED International Conference CSN*, 2005, pp.224 - 229.
- [4] E. R. Berlekamp, *Algebraic Coding Theory*. New York:McGraw-Hill, 1968.
- [5] M. Sudan, "Decoding of Reed-Solomon codes beyond the error-correction bound", *J. Complexity*, vol. 13, no. 1, pp. 180 - 193, 1997.
- [6] R. M. Roth and G. Ruckenstein, "Efficient decoding of Reed-Solomon codes beyond half the minimum distance," *IEEE Trans. Inform. Theory*, vol. 46, pp. 246-257, Jan. 2000.
- [7] S. L. Goff, A. Glavieux, and C. Berrou, "Turbo-codes and high spectral efficiency modulation", in *Proc. IEEE Int. Conf. Commun.*, 1994, pp.645 - 649.
- [8] P. Jung, "Novel low complexity decoder for turbo codes". *Electron. Lett.*, pp. S6 - S7, Jan. 1995.
- [9] E. K. Hall and S. G. Wilson, "Design and Analysis of Turbo Codes on Rayleigh Fading Channels", *IEEE J. Select. Areas in Commun.*, vol. 16, No. 2, pp. 160 - 174, Feb. 1998.
- [10] Jilei Hou, P. H. Siegel, and L. B. Milstein, "Performance Analysis and Code Optimisation of Low Density Parity-Check Codes on Rayleigh Fading Channels", *IEEE J. Select. Areas in Commun.*, vol. 19, No. 5, pp. 924 - 934, May, 2001.
- [11] A. J. Viterbi, "A robust ratio-threshold technique to mitigate tone and partial band jamming in coded MFSK systems, " in *IEEE Military Commun. Conf. Rec.*, Oct. 1982, pp22.4.1-22.4.5.
- [12] C. W. Baum and M. B. Pursley, "Bayesian methods for erasure insertion in frequency-hop communications with partial-band interference," *IEEE Trans. Commun.*, vol. 40, pp. 1231 - 1238, July 1992.
- [13] F. J. MacWilliams and N. J. A. Sloane, "The Theory of Error Correcting Codes." North-Holland 1977.
- [14] Xiao-Yu Hu, Eleftheriou, E. Arnold, D.M., "Regular and irregular progressive edge-growth Tanner graphs". *IEEE Tran. Inform. Theory*, vol. 51, pp.386 - 398, Jan. 2005.
- [15] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel." *IEEE Tran. Inform. Theory*, vol. 48, pp. 1570 - 1579, June 2002.

# Advances in Iterative Decoding and Maximum Likelihood Decoding for the Packet Network with Comparisons to Fountain Codes over the Erasure Channel

J. Cai, M. Tomlinson, C. Tjhai, M. Ambroze, and M. Ahmed

Fixed and Mobile Communications Research

University of Plymouth

PL4 8AA, United Kingdom

email: {jcai,ctjhai,mtomlinson,mambroze,mahmed}@plymouth.ac.uk

## Abstract

In this paper, we propose a novel ML decoding algorithm – *the In-place Algorithm* in conjunction with a *Product Packetisation* method for the congested Internet Network modeled as a Binary Erasure Channel (BEC). Any code can be used with this algorithm and we give results for cyclic codes constructed from BCH codes. Existing codes and decoding algorithms are compared in terms of performance and decoding complexity. It is shown that a significant performance improvement can be achieved. In general, ML decoding on the Erasure channel has a complexity of  $O(N^3)$  or  $O(N^2)$  depending on the algorithm. It is shown that this complexity can be reduced to  $O(N^{1.5})$  for the network channel by using the product packetisation method. With an analysis in performance, ratelessness can be achieved by acks.

## 1 Introduction

Network transmission is based on packet transmission. Multicast and broadcast are typical examples of packet-based communications. One simple way to get around packet loss is to employ a protocol in which receiving parties acknowledge received packets. This solution requires multiple rounds of communications which is inappropriate for many situations, especially in multicasting and broadcasting. In those cases, we must limit the amount of feedback to the senders and the number of redundant packets sent to receivers. Erasure codes rely on error correcting principle which can recover the lost packets at the receiver without the need for retransmission.

A model for the Internet transmission is the Binary Erasure Channel (BEC). This channel was introduced by Elias [2] in 1955. A packet is lost due to the network congestion with probability  $p$ , and then the BEC has the capacity  $1 - p$ . Elias proved that there exist codes of rate  $R$  for any  $R < 1 - p$  that can be used to transmit over channels of capacity  $1 - p$ . For an erasure channel a rate  $R$  code encodes a message of size  $K$  into a transmission of  $K/R = N$ , so that the original message can be recovered from erased positions of the  $N$  bit message.

In a milestone paper, Luby *et. al.* [3] proposed the first realization of a class of erasure codes – LT codes, which are termed rateless and are generated on the fly as needed. In [4], Shokrollahi introduced the idea of Raptor codes which adds an outer code to LT codes. Raptor codes have been established in order to solve

the error floors exhibited by the LT codes.

In this paper we consider stand alone BCH codes instead of LT codes, as they do not show an error floor. Low-density parity-check (LDPC) codes have been studied [1] to [9] for application to the BEC. Gallager's soft-decoding algorithm [10], was implemented [1] for LDPC codes over the BEC. Capacity-achieving degree distribution codes for the binary erasure channel have been introduced in [1], [7] and [8]. Finite-length analysis was used to find better LDPC codes for the BEC [9]. For ML decoding on the BEC, three families of algorithms have been proposed: structured Gaussian elimination [12], whose computational complexity is  $O(N^3)$ , the conjugate gradient and Lanczos algorithms [12] and [13], both of which require about  $O(N^2)$  operations, and the Wiedemann [14] algorithm with  $O(N^2)$ . The Lanczos algorithm incorporating structured Gaussian elimination was proposed in [15].

The paper is organized as follows. Starting with a superposition of the erased bits on the parity-check matrix in section 2.1, we review the performance of the iterative decoding algorithms, for the BEC, in section 2.2, principally the Recovery Algorithm and the Guess Algorithm and an improved algorithm [6] – the Multi-Guess Algorithm. In section 2.3, we also describe a Maximum Likelihood decoding algorithm, the In-place Algorithm, which is based upon the Gaussian Elimination Algorithm [12]. In section 3, we propose a new packetisation format – A Product Packetisation, which provides the basis for a more efficient and quicker decoder. Using the In-place algorithm, the computational complexity is  $O(N^{1.5})$ . The conclusions are given in

## 2 Preliminaries

### 2.1 Matrix Representation of erased bits

We use the parity check matrix  $H$  to represent the code. Considering an  $(N, K)$  binary linear block code, we denote a codeword as  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ . After being transmitted over the erasure channel with erasure probability  $\epsilon$ , the received vector can be divided into a transmitted sub-sequence and an erased sub-sequence, denoted as  $\mathbf{y} = \{y_1, y_2, \dots, y_{L_r}\}$  and  $\mathbf{y}_\epsilon = \{y_{\epsilon 1}, y_{\epsilon 2}, \dots, y_{\epsilon L_\epsilon}\}$  respectively, where  $L_r + L_\epsilon = N$ .

Corresponding to the parity check matrix of the code, we can generate an erasure matrix  $M_\epsilon (L_\epsilon \times N)$  which contains the positions of the erased bits in  $H$ . Then we denote the set of erased bits  $i$  that participate in each parity check row by  $E_i^h = \{j : M_{\epsilon(ij)} = 1\}$  with  $h$  standing for "horizontal" and the number of erased bits in  $E_i^h$  is denoted by  $|E_i^h|$ . Similarly we define the set of checks in which bit  $j$  participates,  $E_j^v = \{i : M_{\epsilon(ij)} = 1\}$  with  $v$  standing for "vertical", and the number of erased bits in  $E_j^v$  is denoted by  $|E_j^v|$ . Let  $\mathbf{E}^h = \{E_i^h \mid i \in \{1, 2, \dots, L_\epsilon\}\}$  and  $\mathbf{E}^v = \{E_j^v \mid j \in \{1, 2, \dots, N\}\}$ . The matrix representation is shown in Fig. 1, where an "x" represents an erasure.

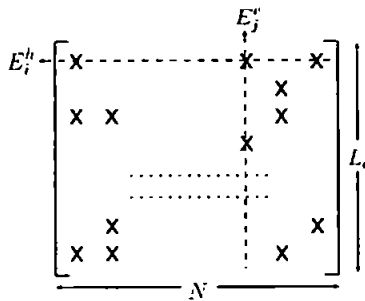


Fig. 1. A matrix representation ( $M_\epsilon$ ) of the erased bits

### 2.2 Review of Iterative Decoding Algorithms

In [1], the Recovery Algorithm is described, which is also called message-passing algorithm, and is equivalent to Gallager's soft-decoding algorithm, an iterative decoding algorithm. The decoding complexity can be shown to be  $O(N^2)$ . The algorithm is briefly outlined below:

#### Recovery Algorithm

- Step 1* Generate the  $M_\epsilon$  and obtain the  $\mathbf{E}^h$ .
- Step 2* For  $i \in \{1, 2, \dots, L_\epsilon\}$ , if  $|E_i^h| = 1$ , we replace the value in the bit position  $i$  with the XOR of the unerased bits in that check equation. Then we remove the erasure from the erasure matrix.

- Step 3* Continue from step 2 until all the erased bits are solved or the decoding cannot continue further.

If there exists erasures in stopping sets, the Recovery Algorithm stalls. In this case, an algorithm [11] with the term Guess Algorithm is applied by performing several "guesses" of the unsolved erased bits, which is similar to the Maxwell decoder [19] which works as a message passing decoder.

#### Guess Algorithm

- Step 1* Run the decoder with Recovery Algorithm until it fails due to stopping set(s).
- Step 2* In order to break the stopping set, when  $|E_i^h| = 2$ , we guess one of the erased symbols and update the erasure matrix  $M_\epsilon$  and  $\mathbf{E}^h$ .
- Step 3* Continue from step 1 until all the erased symbols are solved or the decoding cannot continue further. If the decoder cannot continue, declare a decoder failure and exit.
- Step 4* Create a list of  $2^g$  solutions, where  $g$  is the number of guesses made. From the list  $c_{\text{out},k}$ ,  $k \in \{1, 2, \dots, 2^g\}$ , pick the one that satisfies  $Hc_{\text{out},k}^T = 0$ .

Obviously, compared to the Recovery Algorithm, the complexity of this algorithm increases with  $g$ . Usually, we limit the number of guesses to a small number  $g_s$ . If after  $g_s$  guesses, the decoding still cannot be finished, a decoding failure is declared.

In [6], we proposed an improved iterative decoding algorithm, which is called "Multi-Guess Algorithm". For sparse codes, the Multi-Guess Algorithm can approach optimal performance with significant improvements. Consider the computational complexity, we calculate the minimum number of guesses, denoted by  $\min(g)$  before the decoding. The calculation lemma was given in [6]. For the Multi-Guess Algorithm, a whole row is guessed. A crucial row  $c$  is defined as follows:

- 1)  $c \in \omega_\delta$ , where  $\omega_\delta$  is the set of all equations with  $|E_i^h| = \delta$  where  $i \in \{1, 2, \dots, L_\epsilon\}$ , and  $\delta =$  number of unknown positions.
- 2)  $\sum_{j \in E_i^h} |E_j^v|$  is maximized over  $c$  in  $\omega_\delta$

The Multi-Guess Algorithm is given below:

#### Multi-Guess Algorithm

- Step 1* Run the decoder with Guess Algorithm until  $|E_i^h| > 2$  for  $i = 1, \dots, L_\epsilon$ .
- Step 2* Evaluate the value of  $\min(g)$ . If  $\min(g) > g_s$ , the decoding declares a failure and exits.
- Step 3* Group the rows as  $\omega_\delta$ .
- Step 4* Find the row which is on the basis of the highest value of  $|E_j^v|$  with the value of  $|E_i^h| = \delta$  and guess all erased bits in that row. (There will be at most  $2^{\delta-1}$  guesses.)
- Step 5* Guess one bit  $p$  with  $|E_p^v| = 1$  in each of the independent rows.
- Step 6* Update  $M_\epsilon$ ,  $\mathbf{E}^h$  and  $\mathbf{E}^v$ . Continue the decoding

from step 3 to step 5 until all the erased bits are solved or the decoding cannot continue further.

Obviously, the decoding complexity can grow exponentially with the number of guesses. In practice, improved performance is achieved with  $O(N^2)$  complexity. With higher complexity, there is a better tradeoff with respect to performance by using the ML decoding algorithm.

### 2.3 Optimal ML Decoding Algorithm

Let  $x'$  denote the received vector, where  $x' = y \cup y_e$ . We now devise a reduced complexity algorithm to decode the erased bits by solving the equation 1 using the Gaussian Reduction method [12].

$$Hx'^T = 0. \quad (1)$$

According to [9], optimal decoding is equivalent to solving the linear system, shown in equation (1). If the equation (1) has a unique solution, an optimal algorithm is possible. Gaussian Reduction algorithm is considered as an optimal algorithm over the BEC, but has a complexity of  $O(N^3)$ . We propose a reduced complexity Gaussian Reduction algorithm – In-place Algorithm [5] by eliminating the column-permutations required. This algorithm is stated as follows:

#### In-place Algorithm

*Step 1* The codeword is received and  $y_e$  are substituted in positions of erased bits in  $H$ . Starting with one of the erased symbols,  $y_{e,a}$ , the first equation containing this symbol is flagged that it will be used for the solution of  $y_{e,a}$ . This equation is subtracted from all other equations containing  $y_{e,a}$  and not yet flagged to produce a new set of equations. The procedure repeats until either non flagged equations remain containing  $y_{e,a}$  (in which case a decoder failure is declared) or no erased symbols remain that are not in flagged equations.

*Step 2* Let  $y_{e,last}$  be the erased symbols at the last flagged equations. In the latter case, starting with  $y_{e,last}$  this equation is solved to find  $y_{last}$  and this equation is unflagged. This coefficient is substituted back into the remaining flagged equations containing  $y_{last}$ . The procedure now repeats with the second from last flagged equation now being solved for  $y_{e,last-1}$ . This equation is unflagged and followed by back substitution of  $y_{last-1}$  for  $y_{e,last-1}$  in the remaining flagged equations.

Compared with the Gaussian Elimination Algorithm, the In-place Algorithm complexity remains  $O(N^3)$ , but the multiplicative constant is significantly reduced. The memory required for storing swapped positions is eliminated by only performing flag and addition, which means we do not have to convert the  $M_i$  matrix to a leading diagonal matrix. The least complexity ML algorithm [18] which requires  $O(N^{2.376})$  in complexity

is not guaranteed to produce a solution and incurs considerable overheads in implementation.

## 3 Product Packetisation

In this section, we describe a product packetisation structure. In a traditional way, files sent over the Internet are partitioned into packets, as in the applications of LT codes and Raptor codes. Each packet is either received without error or not in which case it is erased. Standard file-transfer protocols simply partition a file into  $n$  packets, then repeatedly transmit each packet until all of the packets have been received.

The term Product Packetisation is used to describe the arranging of the encoded bits in a format similar to a product code. The encoded bits are written into  $n$  columns and read out in rows for transmission, as illustrated in Fig. 2. With this packetisation structure, short, powerful algebraic codes can be used for Internet transmission.

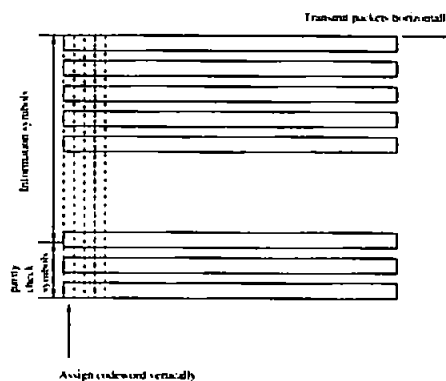


Fig. 2. A Product Packetisation Structure

We designed the structure as a square matrix of size  $n \times n = N$ , where  $N$  is the size of the encoded sequence. Although the structure is flexible, there is a complexity/performance compromise which depends on code block size. We have found that, in this case, choosing a square matrix structure gives a good complexity/performance compromise. In Fig. 3, the shadowed area represents a packet erasure. In the decoding process, each column is decoded separately. As the erasures are in the same positions, for all codewords, the In-place Algorithm has to be applied only to the first column. For each of the other  $n - 1$  columns, the same erasure correcting equations are used, but with the coordinates provided by that column. This reduces the complexity of the ML decoder to  $O(n^3) = O(N^{3/2})$  since  $n = \sqrt{N}$ . This coding scheme can be made rateless by acknowledging successful decoding once enough packets have been received. Due to the powerful error correction codes used, not all the packets are necessary for successful decoding. Compared with fountain codes using the message-passing algorithm



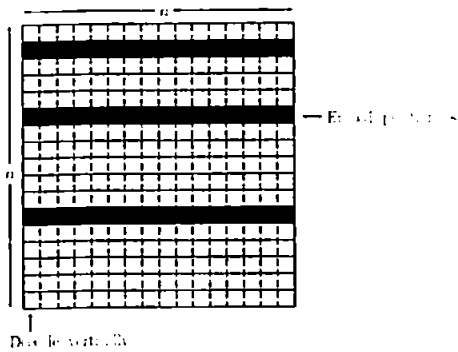


Fig. 3. An example of received packets

with complexity  $O(N^2)$ , the new algorithm is not only able to correct more erasures but is also faster and more efficient.

## 4 Simulation Results

We have evaluated the performance of LT codes and cyclic codes for  $N = 225$  and  $N = 3969$ . For  $N = 225$ , the LT code with the recovery decoder produces very poor performance. In this case, we have used the ML decoder. The LT code (225, 105) was divided into 15 packets with the packet-size of 15 bits and the cyclic code (225, 105) was arranged in the product packetisation format with the same packet-size<sup>1</sup>. The cyclic code is capable of correct decoding up to and including 4 packets erased. As  $1 + x^{15}$  divides  $1 + x^{225}$ , the decoder for this cyclic code can be implemented with the reduced complexity decoder. As shown in Fig 4, the performance of the cyclic code (225, 105) is significantly better than that of the LT (225, 105) code. When  $p = 0.1$ , the result of the cyclic code (225, 105) is over 2 orders of magnitude better than that of the LT (225, 105).

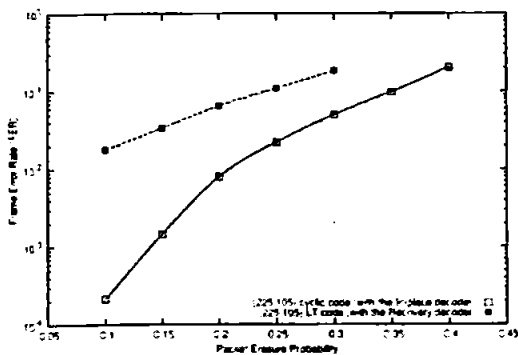


Fig. 4. FER vs the erasure probability of erased packets for the cyclic code (225, 105) and the LT (225, 105) code over the Erasure Channel, both using ML decoding

<sup>1</sup>The cyclic code (225, 105) has a generator polynomial  $g(x) = 1 + x^{15} + x^{30} + x^{60} + x^{120}$ , which is a replication of the BCH (15, 7, 5) code with  $g(x) = 1 + x + x^2 + x^4 + x^5$ .

To verify the decoding complexity calculation, we implemented a longer code, the cyclic code (3969, 2016) and the LT (3969, 2016) code. The cyclic code<sup>2</sup> (3969, 2016) is based upon the generator polynomial of the BCH (63, 32, 12) code. The simulated performance is shown in Fig. 5. On a standard 1GHz PC, the LT (3969, 2016) code took 775ms to encode each codeword and 141ms to decode each codeword for  $p = 0.1$  (using the Recovery Decoder). In contrast, the cyclic code (3969, 2016) took only 5ms to encode each codeword and 8ms to decode each codeword using the In-place decoder for  $p = 0.4$  (the decoding time for this decoder is independent of the value of  $p$ ).

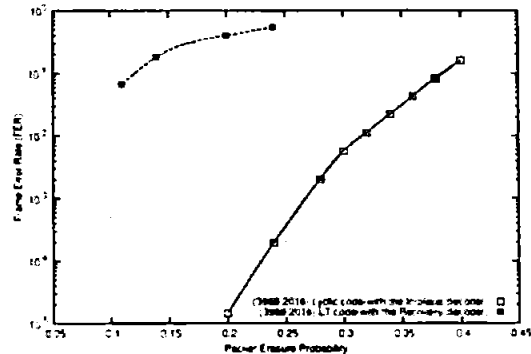


Fig. 5. FER vs the erasure probability of erased packets for the cyclic code (3969, 2016) and the LT (3969, 2016) code over the Erasure Channel

## 5 Conclusions

In this paper, we presented a novel coding and decoding algorithm for packet networks modelled as a Binary Erasure Channel (BEC). The algorithm combines a ML decoder with a product packetisation structure. In the past iterative decoding algorithms, although non-optimum, have been used as a good tradeoff between performance and computational complexity. With the new decoding algorithm, decoding is ML and the computational complexity is reduced to  $O(N^{1.5})$ . This is less complexity than the case for iterative decoders. Also, compared with LT codes, the new decoding algorithm with algebraic codes achieves a significant improvement in performance.

## References

- [1] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman, "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569 – 584, Feb. 2001.
- [2] P. Elias, "Coding for two noisy channels," in *Proc. 3rd London Symp. Information Theory*, 1955, pp.61 – 76.

<sup>2</sup>The cyclic code (3969, 2016) has a generator polynomial  $g(x) = 1 + x^{63} + x^{111} + x^{567} + x^{756} + x^{882} + x^{915} + x^{1008} + x^{1131} + x^{1197} + x^{1260} + x^{1323} + x^{1638} + x^{1761} + x^{1890} + x^{1953}$

- [3] M. Luby, "LT-Code," *Proceeding of the 43rd Annual IEEE Symposium on the Foundations of Computer Science*, pp. 271 – 280, 2002.
- [4] M. A. Shokrollahi, S. Lassen and M. Luby, "Multi-stage code generator and decoder for communication systems," U.S. Patent application number 20030058958, Dec. 2001.
- [5] M. Tomlinson, J. Cai, Cen Tjhai, M. Ambroze and M. Z. Ahmed, "System for correcting deleted or missing symbols," U. K. Patent application number 0428042.6, Dec. 2004.
- [6] J. Cai, Cen Tjhai, M. Tomlinson, M. Ambroze and M. Z. Ahmed, "An Efficient Solution to Packet Loss : Erasure Correcting Codes," in *The Fourth IASTED International Conference CSN*, 2005, pp.224 – 229.
- [7] M. A. Shokrollahi, "Capacity-achieving sequences," *IMA Volumes in Mathematics and its Applications*, vol.123, pp.153 – 166, 2000.
- [8] P. Oswald and M. A. Shokrollahi, "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp.3019 – 3028, Dec. 2002.
- [9] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. Urbanke, "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Tran. Inform. Theory*, vol. 48, pp. 1570 – 1579, June 2002.
- [10] R. G. Gallager, "Low Density Parity Check Codes," Cambridge, MA: MIT Press, 1963.
- [11] H. Pishro-Nik and Faramarz Fekri, "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 20, pp. 439 – 454, March 2004.
- [12] A. M. Odlyzko, "Discrete logarithms in finite fields and their cryptographic significance," in *Advances in Cryptology: Proceeding of Eurocrypt'84*, T. Beth, N. Cot, and I. Ingemarsson, Eds. Berlin, Germany: Springer-Verlag, 1985, vol. 209, pp. 224 – 314.
- [13] D. Coppersmith, A. Odlyzko and R. Schroepfel, "Discrete logarithms in  $GF(p)$ ," *Algorithmica*, vol. 1, pp 1–15, 1986.
- [14] D. Wiedemann, "Solving sparse linear equations over finite fields", *IEEE Trans. Inform. Theory*, vol IT-27, pp. 54–62, Jan. 1986
- [15] P. L. Montgomery, "A block Lanczos algorithm for finding dependencies over  $GF(2)$ ," in *Advances in Cryptology : Proceeding of Eurocrypt'95*, L. Cl Guillou and J. J. Quisquater, eds, vol 921, pp. 106–120, Springer-verlag, 1995.
- [16] W. W. Peterson, "Error Correcting Codes," The MIT Press 1961.
- [17] F. J. MacWilliams and N. J. A. Sloane, "The Theory of Error Correcting Codes," North-Holland 1977.
- [18] D. Coppersmith and S. Winograd, "Matrix multiplication via arithmetic progressions", in *Proc. 19th ACM Symp. Theory of Computing*, 1987, pp. 1–6.
- [19] C. Measson, R. Urbanke, A. Montanari and T. Richardson, "Life Above Threshold: List Decoding to Area Theorem and MSE", in the ITW 2004, San Antonio, Texas, USA, Oct. 2004.

# AN EFFICIENT SOLUTION TO PACKET LOSS : ERASURE CORRECTING CODES

J. Cai

Fixed and Mobile Communications Research  
University of Plymouth  
Plymouth, Devon, United Kingdom  
email: jcai@plymouth.ac.uk

C. Tjhai, M. Tomlinson, M. Ambroze and M. Ahmed

Fixed and Mobile Communications Research  
University of Plymouth  
Plymouth, Devon, United Kingdom  
email: {ctjhai,mtomlinson,mambroze,mahmed}@plymouth.ac.uk

## ABSTRACT

This paper investigates decoding of binary linear block codes over the binary erasure channel (BEC). Of the current iterative decoding algorithms on this channel, we review the Recovery Algorithm and the Guess Algorithm. We then present a Multi-Guess Algorithm extended from the Guess Algorithm and a new algorithm – the In-place Algorithm. The Multi-Guess Algorithm can push the limit to break the stopping sets. However, the performance of the Guess and the Multi-Guess Algorithm depend on the parity-check matrix of the code. Simulations show that we can decrease the frame error rate by several orders of magnitude using the Guess and the Multi-Guess Algorithms when the parity-check matrix of the code is sparse. The In-place Algorithm can obtain better performance even if the parity check matrix is dense. We consider the application of these algorithms in the implementation of multicast and broadcast techniques on the Internet. Using these algorithms, a user does not have to wait until the entire transmission has been received.

## KEY WORDS

Erasure correcting codes, network congestion, multicast and broadcast

## 1 Introduction

The Binary Erasure Channel (BEC) was introduced by Elias [1] in 1955. It counts lost information bits as being “erased” with probabilities equal to 0.5. Currently, the BEC is widely used to model the Internet transmission systems, in particular multicasting and broadcasting.

As a milestone, Luby *et. al.* [2] proposed the first realization of a class of erasure codes – LT codes, which are rateless and are generated on the fly as needed. However, LT-codes cannot be encoded with constant cost if the number of collected output symbols is close to the number of input symbols. In [3], Shokrollahi introduced the idea of Raptor codes which adds an outer code to LT codes. Raptor codes have been established in order to solve the error floors exhibited by the LT codes.

On the other hand, low-density parity-check (LDPC) codes have been studied [5] to [8] for application to the BEC. The iterative decoding algorithm, which is the

same as Gallager’s soft-decoding algorithm [9], was implemented [5]. Capacity-achieving degree distributions for the binary erasure channel have been introduced in [5], [6] and [7]. Finite-length analysis of LDPC codes over the BEC was accomplished in [8]. In that paper, the authors have proposed to use finite-length analysis to find good finite-length codes for the BEC.

In this paper, we show the derivation of a new decoding algorithm to improve the performance of binary linear block codes on the BEC. The algorithm can be applied to any linear block code and is not limited to LDPC codes. Starting with superposition of the erased bits on the parity-check matrix, we review the performance of the iterative decoding algorithms, described in the literature, for the BEC, principally the Recovery Algorithm and the Guess Algorithm [10]. In Section 3, we propose an improvement to the Guess Algorithm based on multiple guesses: the Multi-Guess Algorithm and give a method to calculate the minimum number of guesses required in the decoding procedure. In this section, we also describe a new, non iterative decoding algorithm based on a Gaussian-Reduction method [11] by processing the parity-check matrix. In Section 4, we compare the performance of these algorithms for different codes using computer simulation. In Section 5, we discuss the application of these decoding algorithms for the Internet. Section 6 concludes the paper.

## 2 Preliminaries

### 2.1 Matrix Representations of the Erased Bits

Let  $H$  denote the parity-check matrix. Considering an  $L \times N$  binary linear block code, we assume that the encoded sequence is  $\mathbf{x} = \{x_1, x_2, \dots, x_N\}$ . After being transmitted over the erasure channel with erasure probability  $\epsilon$ , the encoded sequence can be divided into the transmitted sub-sequence and the erased sub-sequence, denoted as  $\mathbf{y} = \{y_1, y_2, \dots, y_{l_1}\}$  and  $\mathbf{y}_\epsilon = \{y_{\epsilon 1}, y_{\epsilon 2}, \dots, y_{\epsilon l_2}\}$  respectively, where  $l_1 + l_2 = N$ .

Corresponding to the parity check matrix of the code, we can generate an erasure matrix  $M_\epsilon (L_\epsilon \times N)$  which contains the positions of the erased bits in  $H$ . Then we denote

the set of erased bits  $i$  that participate in each parity check row by  $E_i^h = \{j : M_{c(ij)} = 1\}$  with  $h$  standing for "horizontal" and the number of erased bits in  $E_i^h$  is denoted by  $|E_i^h|$ . Similarly we define the set of checks in which bit  $j$  participates,  $E_j^v = \{i : M_{c(ij)} = 1\}$  with  $v$  standing for "vertical", and the number of erased bits in  $E_j^v$  is denoted by  $|E_j^v|$ . Let  $\mathbf{E}^h = \{E_i^h \mid i \in \{1, 2, \dots, L_c\}\}$  and  $\mathbf{E}^v = \{E_j^v \mid j \in \{1, 2, \dots, N\}\}$ . The matrix representation is shown in Fig. 1, where an "x" represents an erasure.

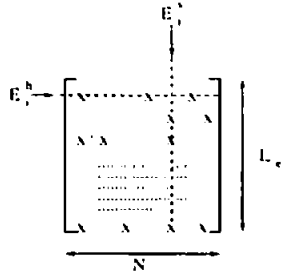


Figure 1. A matrix representation of the erased bits

## 2.2 Current Iterative Decoding Algorithms for the BEC

In [5], the message-passing algorithm was used for reliable communication over the BEC at transmission rates arbitrarily close to channel capacity. The decoding algorithm succeeds if and only if the set of erasures do not cause stopping sets [8]. For completeness, this algorithm is briefly outlined below:

### Recovery Algorithm

- *step 1* Generate the  $M_c$  and obtain the  $\mathbf{E}^h$ .
- *step 2* For  $i \in \{1, 2, \dots, L_c\}$ , if  $|E_i^h| = 1$ , we replace the value in the bit position  $i$  with the XOR of the unerased bits in that check equation. Then we remove the erasure from the erasure matrix.
- *step 3* Continue from step 2 until all the erased bits are solved or the decoding cannot continue further.

The decoder will fail if stopping sets exist.

We can break the stopping sets by performing several "guesses" of the unsolved erased bits. This algorithm is called the Guess Algorithm [10].

### Guess Algorithm

- *Step 1* Run the decoder with Recovery Algorithm until it fails due to stopping set(s).
- *Step 2* In order to break the stopping set, when  $|E_i^h| = 2$ , we guess one of the erased symbols and update the erasure matrix  $M_c$  and  $\mathbf{E}^h$ .

- *Step 3* Continue from step 1 until all the erased symbols are solved or the decoding cannot continue further. If the decoder cannot continue, declare a decoder failure and exit.
- *Step 4* Create a list of  $2^g$  solutions, where  $g$  is the number of guesses made. From the list  $c_{\text{out}k}$ ,  $k \in \{1, 2, \dots, 2^g\}$ , pick the one that satisfies  $Hc_{\text{out}k}^T = 0$ .

Obviously, compared to the Recovery Algorithm, the complexity of this algorithm increases with  $g$ . Usually, we limit the number of guesses to a small number  $g_s$ . If after  $g_s$  guesses, the decoding still cannot be finished, a decoding failure is declared. For sparse codes with low-density  $H$ , e. g. LDPC codes, the Guess Algorithm can improve the performance with  $g < 3$  guesses as shown in Section 4.

The decoding algorithm is more efficient when the bits to be guessed are carefully chosen. These are termed "crucial" bits. The crucial bits are chosen on the basis of the highest value of  $|E_j^v|$  with the value of  $|E_i^h| = 2$ .

## 3 Improved Decoding Algorithms for Non-sparse Linear Block Codes for the BEC

For non-sparse linear codes, it is common to encounter more than 2 unsolved symbols in each row of  $M_c$  after running the Guess Algorithm, due to the high-density of their parity check matrix. In these cases, we cannot break the stopping set by guessing one erased symbol in a row only. More than 1 erased symbols at one time need to be guessed. We can calculate the minimum number of guesses before the decoding.

**Lemma 3.1** Consider the chosen erased symbols in each row as an erased group. Let  $\omega_\delta$  denote the set of rows with  $\delta$  erasures, that is,  $\omega_\delta = \{i \mid |E_i^h| = \delta\}$ . And  $x_\delta$  is the set of rows which satisfies:

$$x_\delta = \{i \in \omega_\delta \mid \exists k, p \in E_i^h, \text{ such as } k \neq p, |E_k^v| = |E_p^v| = 1\}. \quad (1)$$

Then

$$\min g = |x_\delta| + 1 \quad (2)$$

where 1 accounts for the need for at least one "crucial" row.

**Proof 3.1** When the guessing process stops, there are more than 2 erased symbols in each erased row. The rows that have more than two bits  $(k, p)$  which do not participate in any other row (i. e.  $|E_k^v| = |E_p^v| = 1$ ) cannot be solved by other rows, and so at least one of these bits has to be guessed. So the minimum number of guesses equals to the number of all the independent guesses plus one more "crucial" guess to solve the other rows.

For the Multi-Guess Algorithm, a whole row is guessed. A crucial row  $c$  is defined as follows:

$$1. \ c \in \omega_\delta$$

2.  $\sum_{j \in E_p^h} |E_j^h|$  is maximized over  $c$  in  $\omega_\delta$

The Multi-Guess Algorithm is given below:

#### Multi-Guess Algorithm

- *step 1* Run the decoder with Guess Algorithm until  $|E_i^h| > 2$  for  $i = 1, \dots, L_c$ .
- *step 2* Evaluate the value of  $\min(g)$ . If  $\min(g) > g_s$ , the decoding declares a failure and exits.
- *step 3* Group the rows with  $|E_i^h| = \delta$  as  $\omega_\delta$ , where  $i \in \{1, 2, \dots, L_c\}$ .
- *step 4* Find the "crucial" row and guess all erased bits in that row. (There will be at most  $2^{\delta-1}$  guesses.)
- *step 5* Guess one bit  $p$  with  $|E_p^v| = 1$  in each of the independent rows, i.e. the rows in  $x_g$ .
- *step 6* Update  $M_c$ ,  $E^h$  and  $E^v$ . Continue the decoding from step 3 to step 5 until all the erased bits are solved or the decoding cannot continue further.

The disadvantages of Guess and Multi-Guess Algorithms include the decoding complexity and the correctness of the results. The decoding complexity grows exponentially with the number of guesses. It is possible that the group guess declares a wrong value as the result of the decoder. Although this kind of situation happens only when the value of  $\epsilon$  is very small, it is still undesirable.

Let  $x'$  denote the received vector, where  $x' = y \cup y_c$ . We now devise a reduced complexity algorithm to decode the erased bits by solving the equation 3 using the Gaussian Reduction method [11].

$$Hx^T = 0. \quad (3)$$

According to [8], the optimal decoding is equivalent to solving the linear system, shown in the equation 3. If the equation 3 has a unique solution, the optimal algorithm is possible. Gaussian Reduction algorithm is considered as the optimal algorithm over the BEC. We propose a reduced complexity Gaussian Reduction algorithm – In-place Algorithm [4] by eliminating the column-permutations required. This algorithm is stated as follows:

#### In-place Algorithm

- *step 1* The codeword is received and  $y_c$  are substituted in positions of erased bits in  $H$ . Starting with one of the erased symbols,  $y_{c_s}$ , the first equation containing this symbol is flagged that it will be used for the solution of  $y_{c_s}$ . This equation is subtracted from all other equations containing  $y_{c_s}$  and not yet flagged to produce a new set of equations. The procedure repeats until either non flagged equations remain containing  $y_{c_s}$  (in which case a decoder failure is declared) or no erased symbols remain that are not in flagged equations.

- *step 2* Let  $y_{c_{last}}$  be the erased symbols at the last flagged equations. In the latter case, starting with  $y_{c_{last}}$  this equation is solved to find  $y_{last}$  and this equation is unflagged. This coefficient is substituted back into the remaining flagged equations containing  $y_{last}$ . The procedure now repeats with the second from last flagged equation now being solved for  $y_{c_{last-1}}$ . This equation is unflagged and followed by back substitution of  $y_{last-1}$  for  $y_{c_{last-1}}$  in the remaining flagged equations.

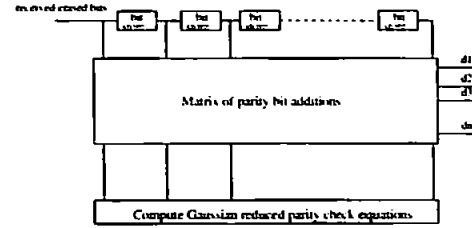


Figure 2. Erasure Correction Using In-place Algorithm

A block schematic of the decoder is shown in Fig.2. The received bits are stored in the shift register with the erased bits being replaced by the unknown  $y_c$ . The Gaussian reduced equations are computed and used to define the connection of bit adders from the respective shift register stage to compute the outputs  $d_1$  to  $d_n$ . The non erased symbols contained in the shift register are switched directly through to the respective output so that the decoded codeword with no erased bits is present at the outputs  $d_1$  through to  $d_n$ .

## 4 Results

We evaluated the performance of the Recovery Algorithm with the LT codes with Soliton distribution as described in [2] and irregular LDPC codes. As shown in Fig. 3, the performance of irregular LDPC codes is significantly better than that of the LT codes for the same block length. As a consequence, we use LDPC codes to benchmark the remaining algorithms.

A particularly strong binary code and which has a sparse  $H$  is the cyclic LDPC code (255,175), which has a length of 255 bits after encoding of 175 information bits. Since the parity-check polynomial of the (255,175) <sup>1</sup> code is orthogonal on every bit position, the minimum Hamming distance is  $1 + w$ , where  $w$  denotes the number of ones per row in  $H$  [12].

The applicability of the decoding methods above depends on the error correcting code being used and specifically on the parity check matrix being used. The performance of this code for the Recovery, the Guess and the In-

<sup>1</sup>The (255,175) Cyclic LDPC code has a minimum Hamming distance of 17.

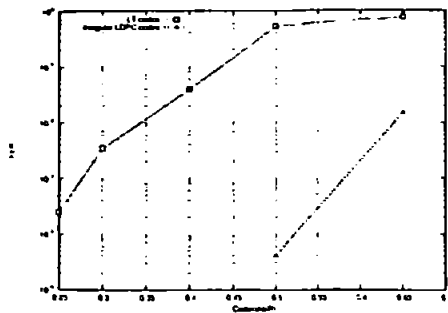


Figure 3. Performance of the LT codes and irregular LDPC codes with erasure probability = 0.2

place Algorithms is shown in Fig. 4 in terms of the probability of decoder error (FER) as a function of the erasure probability for every transmitted bit.

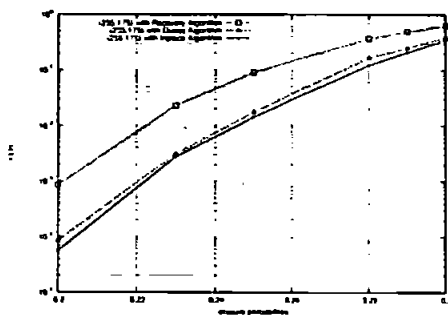


Figure 4. Performance of the Cyclic LDPC (255,175) with the Guess, the Multi-Guess and the In-place Algorithms

Due to its sparse parity check matrix, Guess algorithm with less than 3 guesses can achieve more than 1 order of magnitude improvement compared to that of Recovery Algorithm. In addition, from Fig. 4, we also can see that the curve of Guess Algorithm is very close to the curve of In-place Algorithm, which means Guess Algorithm is a "near optimal decoding" algorithm when it has a sparse parity check matrix.

Fig. 5 shows the performance of the (341,205) LDPC code<sup>2</sup> with the Recovery, the Guess, the Multi-Guess and the In-place Algorithms. Comparing these results of the Recovery and Guess Algorithms, the Multi-Guess Algorithm can obtain the results by several orders of magnitude better. For example, when the erasure probability equals to 0.3, the Multi-Guess Algorithm with  $g_{max} = 3$  is one order of magnitude better than the Recovery and Guess Algorithms, when  $g_{max} = 5$ , the Multi-Guess Algorithm is 2 order2 of magnitude better than the Recovery and the Guess Algorithms. As an optimal decoding algorithm, the In-place Algorithm can achieve 4 orders of magnitude bet-

<sup>2</sup>The (341,105) LDPC code has a minimum Hamming distance of 16.

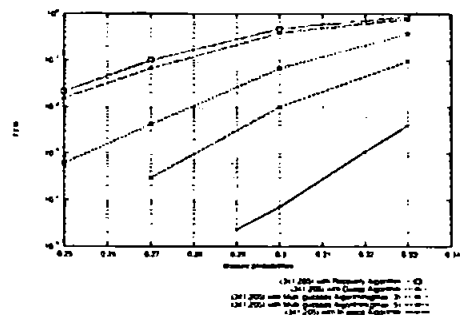


Figure 5. Performance of the Cyclic LDPC (341,205) with the Recovery, the Guess, the Multi-Guess and the In-place Algorithms

ter than the Recovery and the Guess Algorithm.

The ultimate performance of the In-place Algorithm as a function of error correcting code is shown in Fig. 6 for the example (255,175) code which can correct a maximum of 80 erased bits. Fig. 6 shows the probability density function of the number of erased bits short of the maximum correctable which is  $N - L$ . The results were obtained by computer simulations. The probability of being able to correct only 68 bits, a shortfall of 12 bits, is  $1.1 \times 10^{-3}$ . Simulations indicate that on average 77.6 erased bits may be corrected for this code. In comparison the BCH (255,178) code having similar rate is also shown in Fig. 6. The BCH code has a similar rate but a higher minimum Hamming distance of 22 (compared to 17). It can be seen that it has better performance than the (255,175) code but it has a less sparse parity check matrix and consequently it is less suitable for Recovery Algorithm and Guess Algorithm. Moreover the average shortfall in erasures not corrected is virtually identical for the two codes.

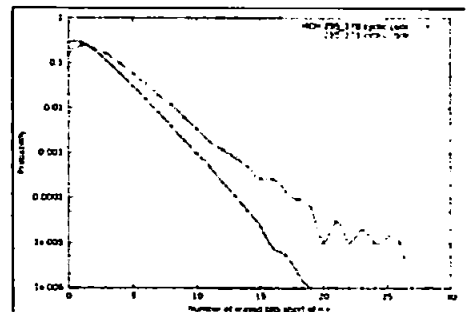


Figure 6. Comparison of Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctable ( $N-L$ ) for (255,175) code and BCH (255,178) code

The simulation results of using In-place Algorithm for the (103,52) quadratic residue binary code [13] are shown in Fig. 7. The minimum Hamming distance for this code is 19 and the results are similar to that of the (255,178)

BCH code above. It is found from the simulations that on average 49.1 erasure bits are corrected (out of a maximum of 51) and the average shortfall from the maximum is 1.59 bits.

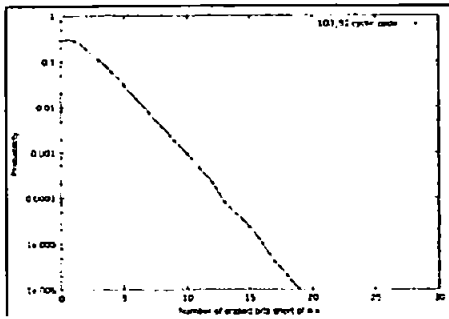


Figure 7. Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctible (N-L) for (103,52) code quadratic residue code

Similarly the results for the extended BCH (128,64) code is shown in Fig. 8. This code has a minimum Hamming distance of 22 and has a similar probability density function to the other BCH codes above. On average 62.39 erasure bits are corrected (out of a maximum of 64) and the average shortfall is 1.61 bits from the maximum.

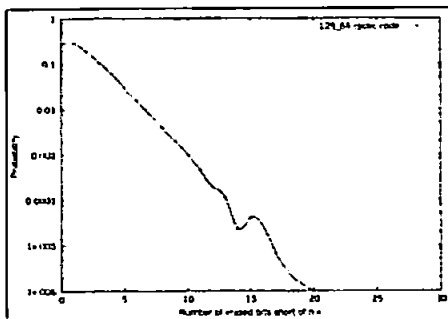


Figure 8. Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctible (N-L) for (128,64) extended BCH code

## 5 Application

In multicast and broadcast information is transmitted in data packets with typical lengths from 30 bits to 1000 bits. These packets could define a symbol from a Galois field [12], viz  $GF(2^m)$  but with  $m$  equal to 30 or more up to and beyond 1000 bits this is impracticable and it is more convenient to use a matrix approach with the packets forming the rows of the matrix and columns of bits encoded using an error correcting code. Usually, but not essentially

the same code would be used to encode each column of symbols. The matrix of symbols may be defined as:

$$\begin{array}{rcl}
 b_{00}b_{01}b_{02}b_{03} \dots b_{0s} & = & \text{packet 1} \\
 b_{10}b_{11}b_{12}b_{13} \dots b_{1s} & = & \text{packet 2} \\
 b_{20}b_{21}b_{22}b_{23} \dots b_{2s} & = & \text{packet 3} \\
 \dots & & \dots \\
 \dots & & \dots \\
 b_{n-10}b_{n-11}b_{n-12}b_{n-13} \dots b_{n-1s} & = & \text{packet n}
 \end{array}$$

There are a total of  $(s + 1) \cdot k$  information symbols which encoded using the parity check equations of a selected code into a total number of transmitted symbols equal to  $(s + 1) \cdot n$ . The symbols are transmitted in a series of packets with each packet corresponding to a row as indicated above. For example the row:  $b_{20}b_{21}b_{22}b_{23} \dots b_{2s}$  is transmitted as a single packet.

Self contained codewords are encoded from each column of symbols. For example  $b_{00}b_{10}b_{20} \dots b_{k-10}$  form the information symbols of one codeword and the remaining symbols,  $b_{k+0}b_{k+10}b_{k+20} \dots b_{n-10}$  are the parity symbols of that codeword. As a result of network congestion, drop outs, loss of radio links or other multifarious reasons not all of the transmitted packets are received. The effect is that some rows above are erased. The decoding procedure is that codewords are assemble from the received packets with missing symbols corresponding to missing packets marked as  $z_{ij}$ . For example, if the second packet only is missing above:

- The first received codeword corresponds to the first column above and is  $b_{00}z_{10}b_{20} \dots b_{n-10}$
- The second codeword corresponding to the first column above and is  $b_{01}z_{11}b_{21} \dots b_{n-11}$  and so on.

All the algorithms stated in Section 2 may be used to solve for the erased symbols  $z_{10}$  in the first received codeword, and for the erased symbol  $z_{11}$  in the second received codeword and so on up to the  $s$ 'th codeword (column) solving for  $z_{1,s-1}$ .

As an example, the binary, extended (128,64) BCH code could be used to encode the information data. The packet length is chosen to be 100 bits, and the total transmission could consist of 128 transmitted packets (12,800 bits total) containing 6,400 bits of information. On average as soon as any 66 packets from the original 128 packets have been received, the remaining 62 packets are treated as if they are erased. 100 codewords are assembled, decoded with the erasures solved and the 6,400 bits of information retrieved. One advantage is that a user does not have to wait until the entire transmission has been received.

## 6 Conclusions

In this paper, we presented different decoding algorithms of LDPC codes over the BEC: Recovery, Guess, Multi-

Guess and In-place Algorithms. The Multi-Guess Algorithm is an extension to Guess Algorithm, which can push the limit to break the stopping sets. We show that Guess and Multi-Guess Algorithms are parity-check matrix dependent. For the codes with sparse parity-check matrix, Guess and Multi-Guess Algorithms can be considered as "Near-optimal Decoding Methods". On the other hand, In-place Algorithm is not. It's an optimal method for the BEC and able to correct  $N - L - \rho$  erasures, where  $\rho$  is a small positive integer.

We also considered these algorithms in the implementation of multicast and broadcast. Using these algorithms, a user does not have to wait until the entire transmission has been received.

## References

- [1] P. Elias. "Coding for two noisy channels." in *Proc. 3rd London Symp. Information Theory*, 1955, pp.61 – 76.
- [2] M. Luby. "LT-Code," *Proceeding of the 43rd Annual IEEE Symposium on the Foundations of Computer Science*, pp. 271 – 280, 2002.
- [3] M. A. Shokrollahi, S. Lassen and M. Luby. "Multi-stage code generator and decoder for communication systems," U.S. Patent application number 20030058958, Dec. 2001.
- [4] M. Tomlinson, J. Cai, Cj Tjhai, M. Ambroze and M. Z. Ahmed. "System for correcting deleted or missing symbols," U. K. Patent application number 0428042.6, Dec. 2004.
- [5] M. Luby, M. Mitzenmacher, M. Shokrollahi, and D. Spielman. "Efficient erasure correcting codes," *IEEE Trans. Inform. Theory*, vol. 47, pp. 569 – 584, Feb. 2001.
- [6] M. A. Shokrollahi, "Capacity-achieving sequences," *IMA Volumes in Mathematics and its Applications*, vol.123, pp.153 – 166, 2000.
- [7] P. Oswald and M. A. Shokrollahi. "Capacity-achieving sequences for the erasure channel," *IEEE Trans. Inform. Theory*, vol. 48, pp.3019 – 3028, Dec. 2002.
- [8] C. Di, D. Proietti, I. E. Telatar, T. Richardson, and R. Urbanke. "Finite-length analysis of low-density parity-check codes on the binary erasure channel," *IEEE Tran. Inform. Theory*, vol. 48, pp. 1570 – 1579, June 2002.
- [9] R. G. Gallager. "Low Density Parity Check Codes," Cambridge, MA: MIT Press, 1963.
- [10] H. Pishro-Nik and Faramarz Fekri. "On decoding of low-density parity-check codes over the binary erasure channel," *IEEE Trans. Inform. Theory*, vol. 20, pp. 439 – 454, March 2004.
- [11] A. M. Odlyzko, "Discrete logarithms in finite fields and their cyptographic significance," in *Advances in Cypology: Proceeding of Eurocrypt'84*, T. Beth, N. Cot, and I. Ingemarssen, Eds. Berlin, Germany: Springer-Verlag, 1985, vol. 209, pp. 224 – 314.
- [12] W. W. Peterson. "Error Correcting Codes," The MIT Press 1961.
- [13] F. J. MacWilliams and N. J. A. Sloane. "The Theory of Error Correcting Codes," North-Holland 1977.



# ANALYSIS ON DECODING ALGORITHMS BASED ON SECTIONALIZED TRELLISES OF BLOCK CODES AND THEIR DUAL

Jing Cai, Marcel Adrian Ambroze  
Digital Communications Research, University of Plymouth, UK  
{J.Cai, M.Ambroze}@plymouth.ac.uk

**Key words to describe this work:** Decoding complexity, Sectionalization, Permutation.

**Key results:** The relationship between permutation-optimal trellis and sectionalization-optimal trellis is found. A new method to count the decoding complexity is provided. The sectionalization method is performed with different decoding algorithms.

**How does the work advance the state-of-the-art?:** The complexity of trellis-based decoding can be reduced by up to 50% with sectionalization.

**Motivation (Problems addressed):** To reduce the computational complexity and the size of memory storage required.

## Introduction

A trellis  $T$  is an edge-labeled directed graph with the property that every state in  $T$  has a well-defined depth which can represent corresponding codes in coding theory. Currently, trellis-based algorithms are widely used, for example, Viterbi algorithm [7] and MAP algorithm [4]. There are many different factors that impose on the complexity of trellises, each decoding method has different level of complexity. In this paper, we briefly survey the complexity of the Viterbi decoding algorithm with Hamming codes and their dual.

## A Review of Sectionalized Trellises

Consider an  $(n, k)$  linear block code  $C$  with a  $n$ -stage bit-level trellis  $T$  in which each branch represents a single code bit. The trellis can be sectionalized by any positive integer  $\nu$  ranging from 1 to  $n$ , so the section boundary set is  $\{h_0, h_1, \dots, h_\nu\}$ , where  $0 = h_0 < h_1 < \dots < h_\nu = n$ . An  $n$ -depth trellis has  $2^{\nu-1}$  sectionalizations, for example, Hamming(7,4) has  $2^6$  different sectionalizations. The main idea in sectionalizing an original trellis is to amalgamate sections that involves two steps: [2].

1. deleting the states and branches between the initial section to the final section;
2. connecting states from the initial section to the final section with the combined labels.

According to [1], we can compute the number of branches  $|B_j|$  and the number of states  $|S_j|$  for any section from the generator matrix of  $C$ .

## Viterbi decoding based on the Sectionalized Trellis

The Viterbi algorithm is a maximum likelihood decoding method which chooses a codeword having the *maximum likelihood metric*, or the *minimum distance metric*.

This decoding procedure consists of two major steps: computing the branch metrics and finding the survivor

paths in the resulting trellis. The decoding implementation includes two metrics: *branch metric* and *state metric*.

Vardy complexity algorithm [1] considers the complexity of computing the branch metric and the complexity of Viterbi decoding separately.

Branch metric complexity means computing the number of operations required in all the branches in one section. Decoding complexity  $|D(T)|$  is the number of operations required to decode the trellis  $T$ . We denote the subcode of  $C$  as  $C(T)$ , and the dual subcode as  $C(T)^\perp$ . There are lots of pre-computations in every step, which can not be easily ignored in the real implementation. In the branch metrics, whether  $C(T)$  is self-complementary need justifying. For an  $N$ -length block code, there are  $NC_i = \sum_{i=1}^N i$  different  $C(T)$  need to be judged. Assume in each section, there are  $m$  codewords. So every  $C(T_j)$  need at most

$$J(T_j) = (m-1) \cdot m + \sum_{i=1}^{m-2} i \quad (1)$$

comparisons. If we pre-store the self-complementary table with the rows meaning the beginning boundary and the columns meaning the ending boundary,  $N \times N$  matrix is required. For example, Hamming (7, 4) need  $NC_i = 27$ ,  $J(T_i) = 77$  and the size of memory storage,  $M_C(T)$  is  $7 \times 7$ .

Also in the branch metrics, we need to consider whether  $1 \in C(T)^\perp$  and the length of the section  $l_i = 0 \pmod 2$  or not. The dual code of  $C(T)$  can be obtained from  $H$  matrix of  $C(T)$ . First, to find all the possibilities of  $l_i = 0 \pmod 2$ , the number of the judgements is denoted by  $Y$ .

$$Y = \sum_{j=0}^{(\text{int})N/2} (N-1) - (2 \times j) \quad (2)$$

And at each section, there are at most  $m$  judgements required to determine whether  $1 \in C(T)^\perp$  or not. The

pre-table requires  $N \times N$  matrix. We still use Hamming(7, 4) as the example, which need  $m = 8$ ,  $Y = 12$ , and memory storage  $7 \times 7$ .

In the decoding part, all the subcodes of  $C$  are required to be considered. There are at most  $\sum_{i=1}^N i \times m$  comparisons for each section. The results also can be pre-stored in an  $N \times N$  matrix. The number of comparisons for the Hamming(7, 4) code is 216.

From the analysis above, we can conclude that the longer the code length, the more calculations and memory storage required, so the complexity of pre-determining calculations can not be simply ignored. So we provide the *Straightforward algorithm*, which trades complexity for implementation simplicity. Recalling the Viterbi decoder, suppose  $S_{h_i, p_j}$  in which  $h_i$  means the current depth and  $p_j$  means the position in this depth,  $D_n$  represents the state and  $O_n$  is the corresponding output. For example, the first stage of branch metric computation for Hamming (7, 4) trellis, we can get the equation as follows:

$$\begin{aligned} S'_{1,1} &= S_{1,1} + (D_1 - O_1)^2 + (D_2 - O_2)^2 + (D_3 - O_3)^2 \\ S'_{1,2} &= S_{1,2} + (D'_1 - O_1)^2 + (D'_2 - O_2)^2 + (D'_3 - O_3)^2 \\ S_{2,1} &= \min\{S'_{1,1}, S'_{1,2}\}. \end{aligned} \quad (3)$$

As described in [3, 5], we consider the number of additions and the number of comparisons as the complexity of Viterbi decoding. Because of the linear property, the number of operations can be obtained section by section. For each section, the number of addition is equal to the number of branches in this section and the number of comparisons is  $|B| - |S_{next}|$ . The Vardy's algorithm is shown as follows:

$$D_v(T_{h,h'}) = 2 \times |B| - |S_1|. \quad (5)$$

Obviously, as the number of stages per section increases, the number of labels per branch will increase. We also need storage to save the labels. So we get the update metric as follows

$$D_v(T_{h,h'}) = 2 \times |B| - |S_1| + |B| \times |\sigma| \quad (6)$$

where  $\sigma$  is the number of labels per branch. Although the complexity is higher than Vardy's method, most criteria of Vardy's depend on lots of comparisons which are ignored in this algorithm, comparisons cannot be simply ignored in real implementation especially for long block codes.

### Optimal sectionalization and Optimal Permutation

Given a code, there are very many different trellises that represent it, of widely varying complexity. When the permutation-optimal trellis is chosen, the decoding complexity can be minimized among all the representations. A permutation that yields the smallest state space dimension at every time of the code trellis and the smallest overall branch complexity is called an optimum permutation [6]. For Hamming codes, the permutation-optimal trellises can normally be obtained directly by natural lexicographic  $H$  matrix. In this case, we use Golay (24,12,8) and count the operations with the straightforward algorithm as the example shown in table 1.

Bit-level trellis operations	Optimum sectionalization	
	operations	boundary location
4472	2558	{0,8,12,14,15,16,24}
89560	5630	{0,11,24}
120904	4478	{0,9,24}

Table 1: Comparison between permutation-optimal and sectionalization-optimal on Golay (24,12,8)

### Conclusion

In this paper, the Viterbi algorithm has been modified with sectionalized trellises. Results of Hamming codes and their dual codes show that the sectionalization method can reduce the computing complexity and the memory storage. With the sectionalization algorithm, the computation complexity can be reduced by nearly 50%. Considering the large number of pre-calculations and large memory storage requirements by the previous metric, we investigated and provided the update metric: *Straightforward algorithm*, which can search out the optimal sectionalization of  $C$  more efficiently and quickly than Vardy's algorithm. And the relation between sectionalization and permutation is found in the paper. It turns out that the sectionalization-optimal code with the permutation-optimal mode can drastically change the number of the computational operations in decoding procedure, often by an exponential factor.

### References

- [1] A. Vardy: Trellis Structure of Codes, Handbook of Coding Theory, (edited by V. S. Pless, W. C. Huffman, and R. A. Brualdi), Elsevier Science Publishers, 1998.
- [2] A. Lafourcade and A. Vardy, "Optimal Sectionalization of a Trellis," *IEEE Trans. on Information Theory*, 42, 3, pp. 689-703, May 1996.
- [3] Robert J. McEliece, "On the BCJR Trellis for Linear Block Codes," *IEEE Trans. on Information Theory*, vol. 42, 4, pp. 1072-1091, July 1996.
- [4] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal Decoding of Linear Codes for Minimizing Symbol Error Rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, 1974.
- [5] A. Vardy and F. R. Kschischang, "Proof of a conjecture of McEliece regarding the expansion index of the minimal trellis," *IEEE Trans. on Information Theory*, vol. 42, pp. 2027-2033, 1996.
- [6] S. Lin, T. Kasami, T. Fujiara, M. Fossorier, *Trellises and Trellis-Based Decoding Algorithm for Linear Block Codes*, Kluwer Academic Publisher, 1998.
- [7] Viterbi, A. J., "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Trans. Inform. Theory*, IT-13, pp. 689-703, 1967.

(12) UK Patent Application (19) GB (11) 2 421 599 (13) A

(43) Date of A Publication 28.06.2006

(21) Application No: 0428042.6  
(22) Date of Filing: 22.12.2004

(71) Applicant(s):  
Martin Tomlinson  
The Old Coach House, Tristford,  
Harberton, TOTNES, Devon, TQ9 7RZ,  
United Kingdom

Jing Cai  
No.13A, North Road East, PLYMOUTH,  
Devon, PL4 6AS, United Kingdom

(continued on next page)

(51) INT CL:  
G06F 11/08 (2006.01) G06F 11/10 (2006.01)

(52) UK CL (Edition X ):  
G4A AEE APCH

(56) Documents Cited:  
EP 0903955 A1 US 5115436 A  
US 4555784 A

(58) Field of Search:  
UK CL (Edition X ) G4A  
INT CL G06F  
Other:

(54) Abstract Title: Correction for missing or deleted symbols using parity check equations

(57) In many communications systems, broadcasting systems and storage systems, information is encoded as a sequence of symbols. The total number of symbols transmitted or stored is denoted by  $n$ , and these are obtained by encoding  $k$  information symbols using parity check equations from an error correcting code (ECC). The present invention allows the  $k$  information symbols to be recovered before all  $n$  symbols have been received or read, thereby compensating for missing symbols and allowing faster access to the information. Three decoding methods using parity check equations to recover missing symbols are described, each of a different degree of complexity. It is shown that the present invention can cope with a higher number of missing symbols than the current state of the art.

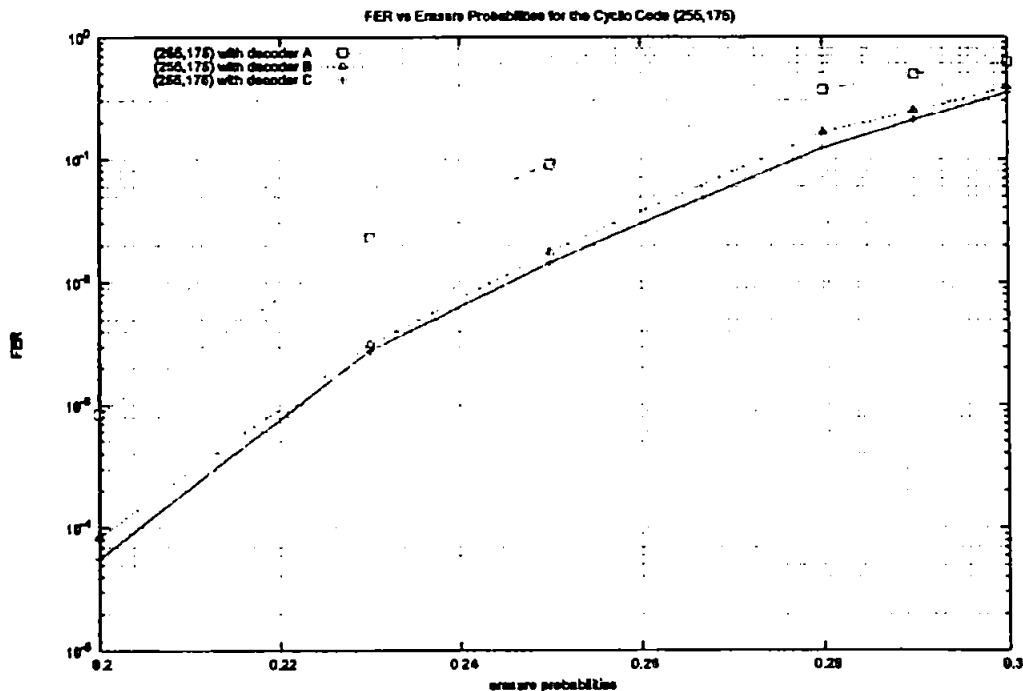


Fig 5 Performance of Methods A,B and C as a function of code and erasure bit probability

**GB 2421599 A continuation**

**(71) cont**

**C.J Tjhai  
Room A201B, St. Thomas Court,  
20 Gasking Street, PLYMOUTH, PL4 9AP,  
United Kingdom**

**Marcel Arabroth  
4B Pier Street, PLYMOUTH, PL1 3BS,  
United Kingdom**

**Mohammed Zaiki Ahmed  
226 Ladysmith Rd, PLYMOUTH, PL4 7NR,  
United Kingdom**

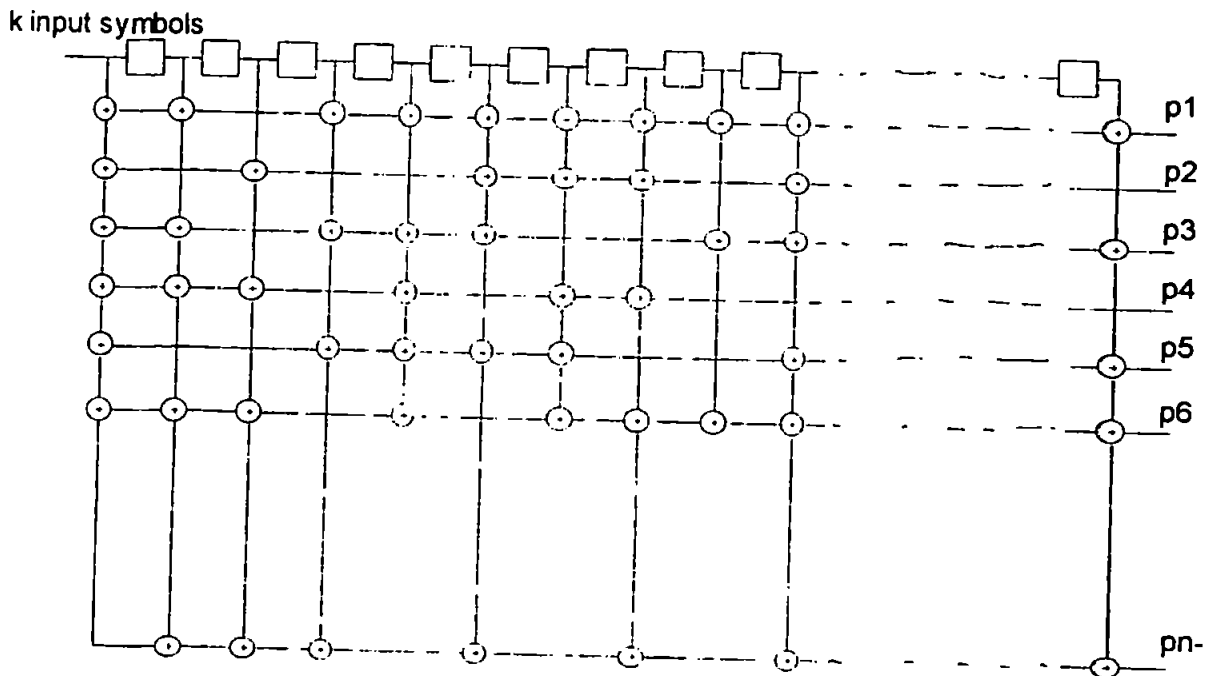
**(72) Inventor(s):**

**Martin Tomlinson  
Jing Cai  
C.J Tjhai  
Marcel Arabroth  
Mohammed Zaiki Ahmed**

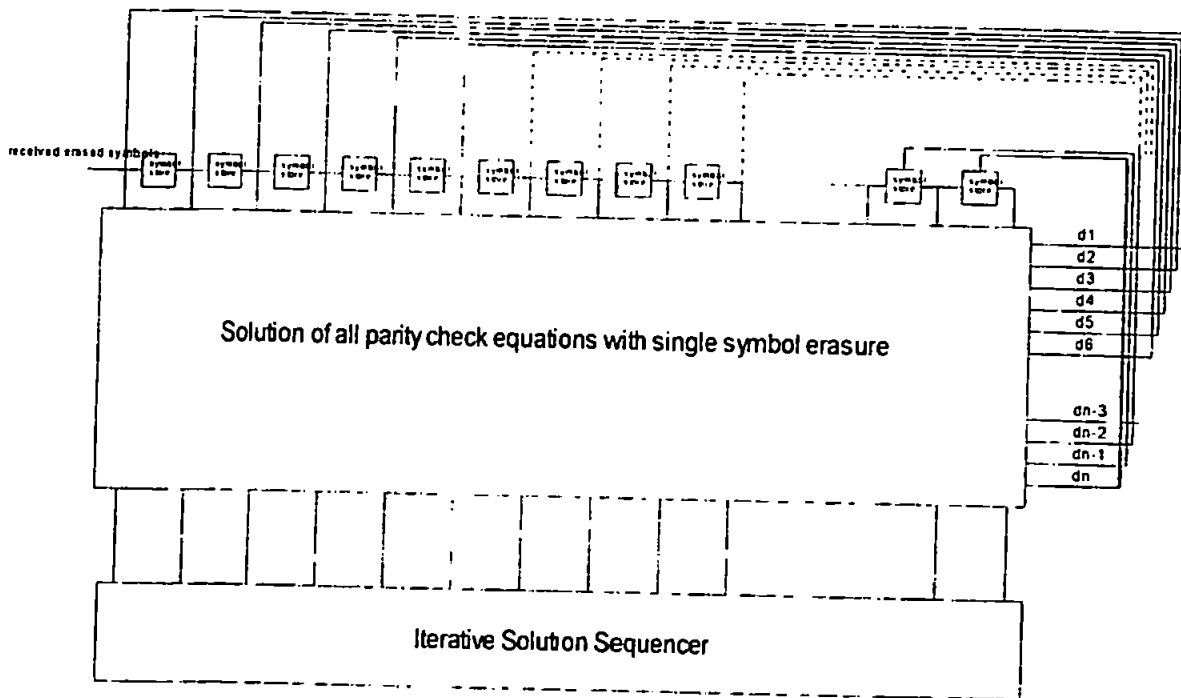
**(74) Agent and/or Address for Service:**

**Martin Tomlinson  
The Coach House, Tristford, Harberton,  
TOTNES, Devon, TQ9 7RZ,  
United Kingdom**

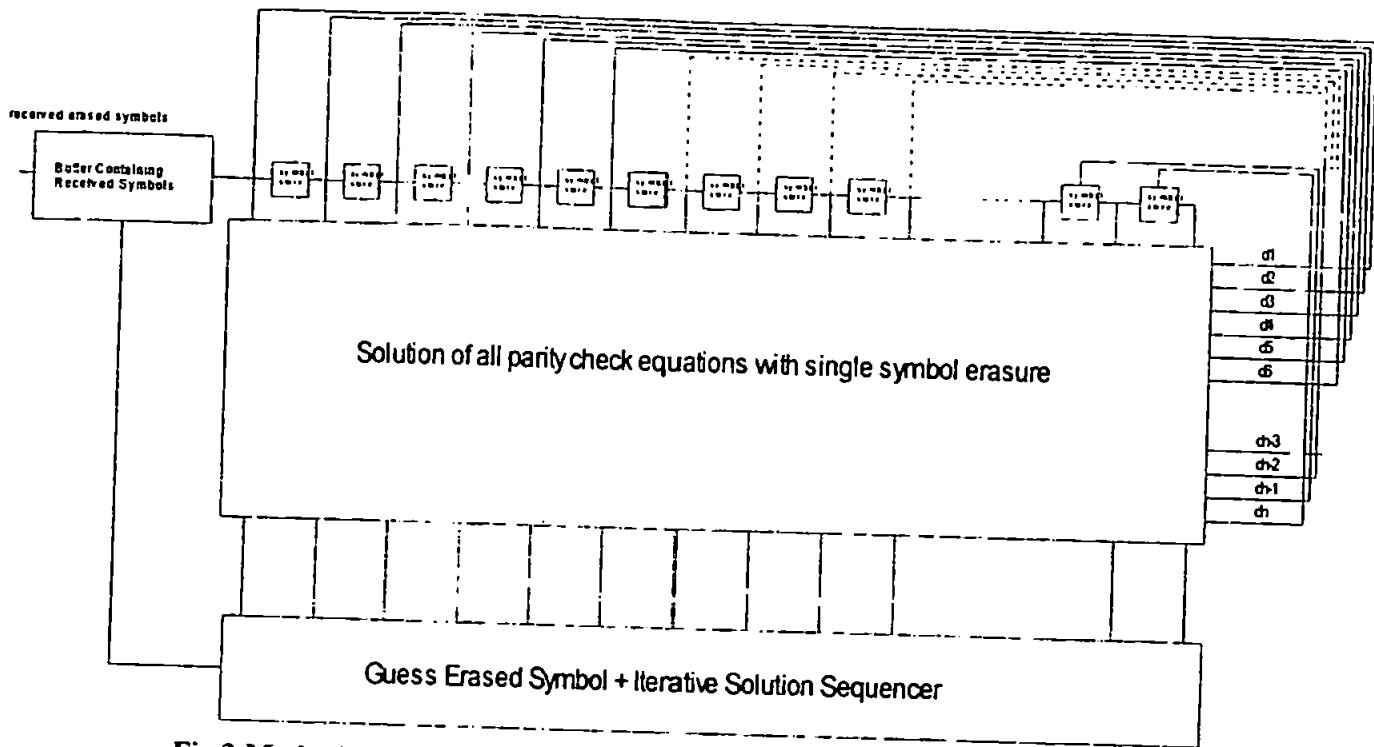
**DIAGRAMS**



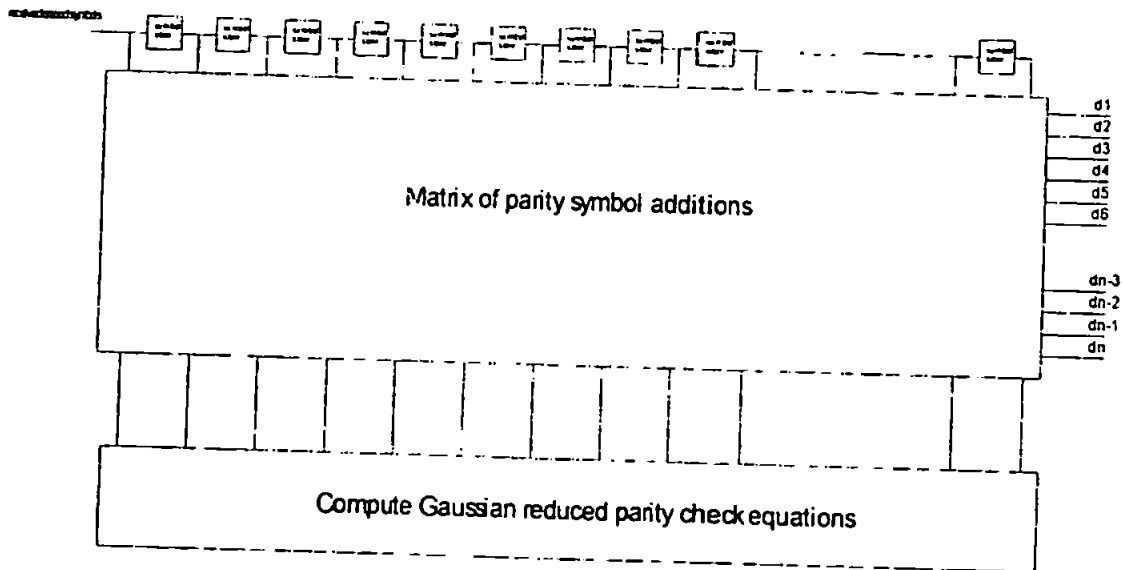
**Fig 1 Encoding of  $n-k$  parity symbols from input  $k$  symbols**



**Fig 2 Method A: Correction of Erased Symbols Using Sequential Solution of Single Erased Symbol Parity Check Equations**



**Fig 3 Method B: Correction of Erased Symbols Using Guess Erased Symbols + Sequential Solution of Remaining Single Erased Symbol Parity Check Equations**



**Fig 4 Method C: Erasure Correction using Gaussian Reduction and Solution of Parity Check Equations**

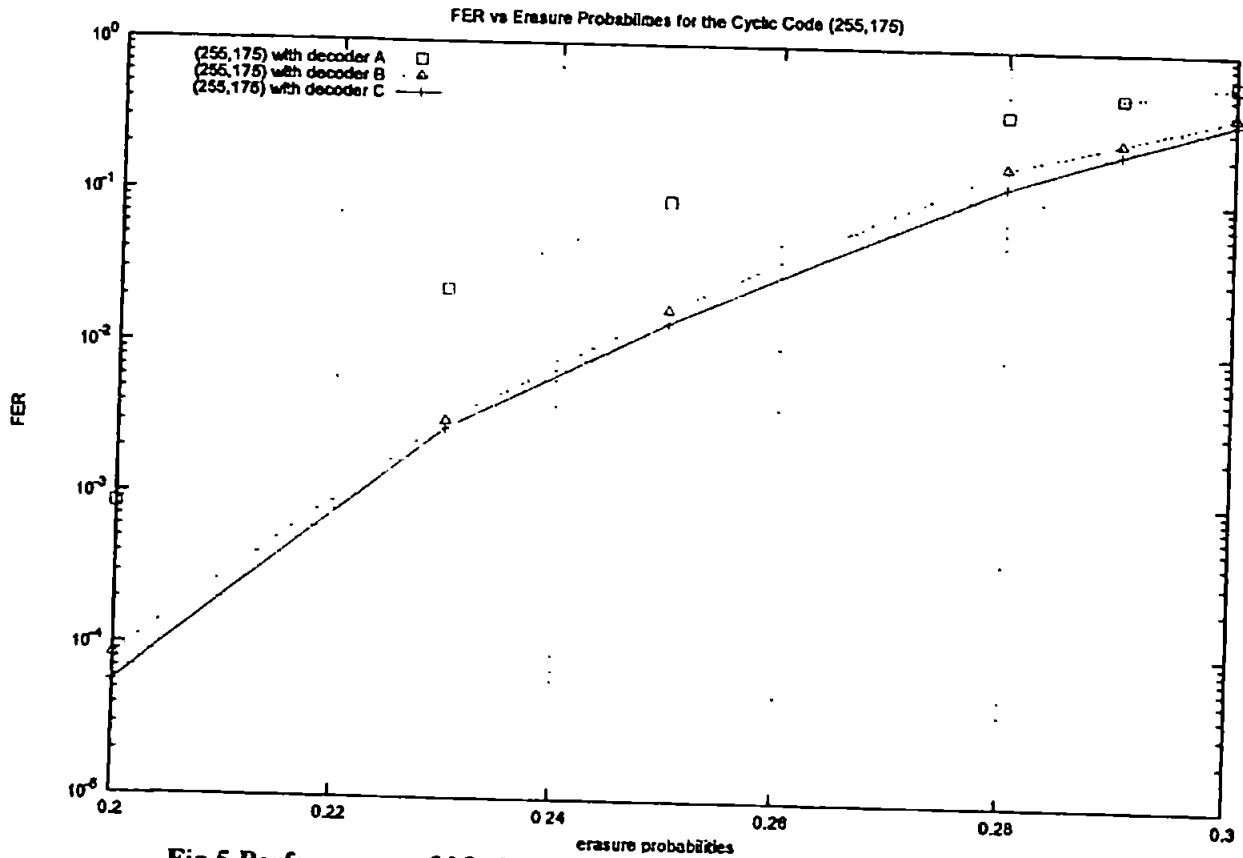


Fig 5 Performance of Methods A,B and C as a function of code and erasure bit probability

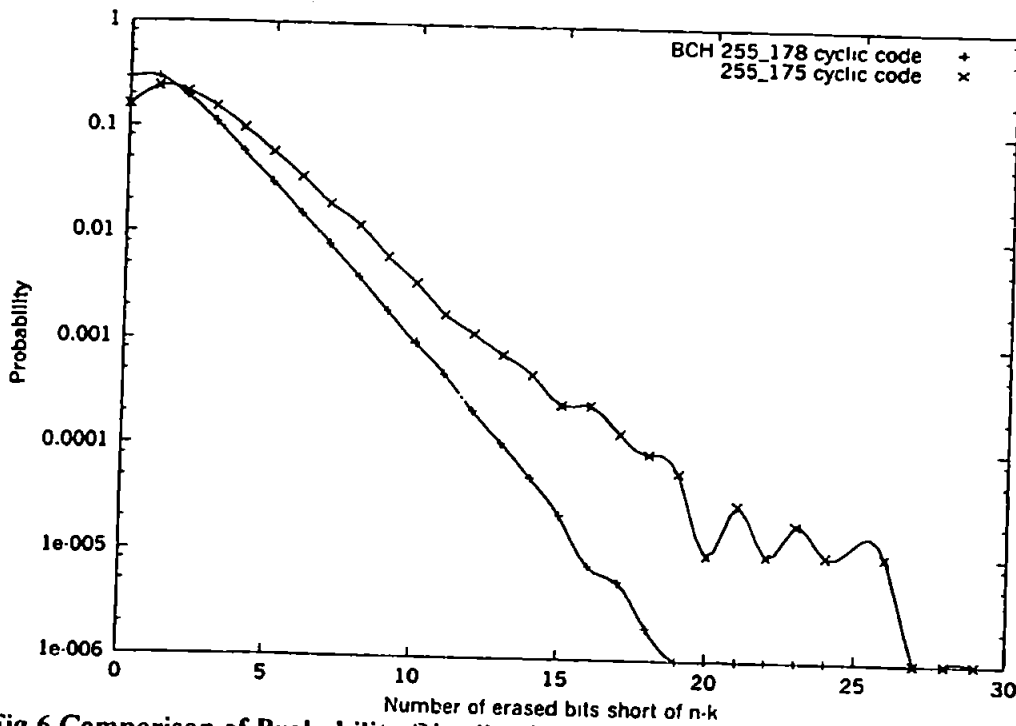


Fig 6 Comparison of Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctible (n-k) for (255,175) code and BCH (255,178) code

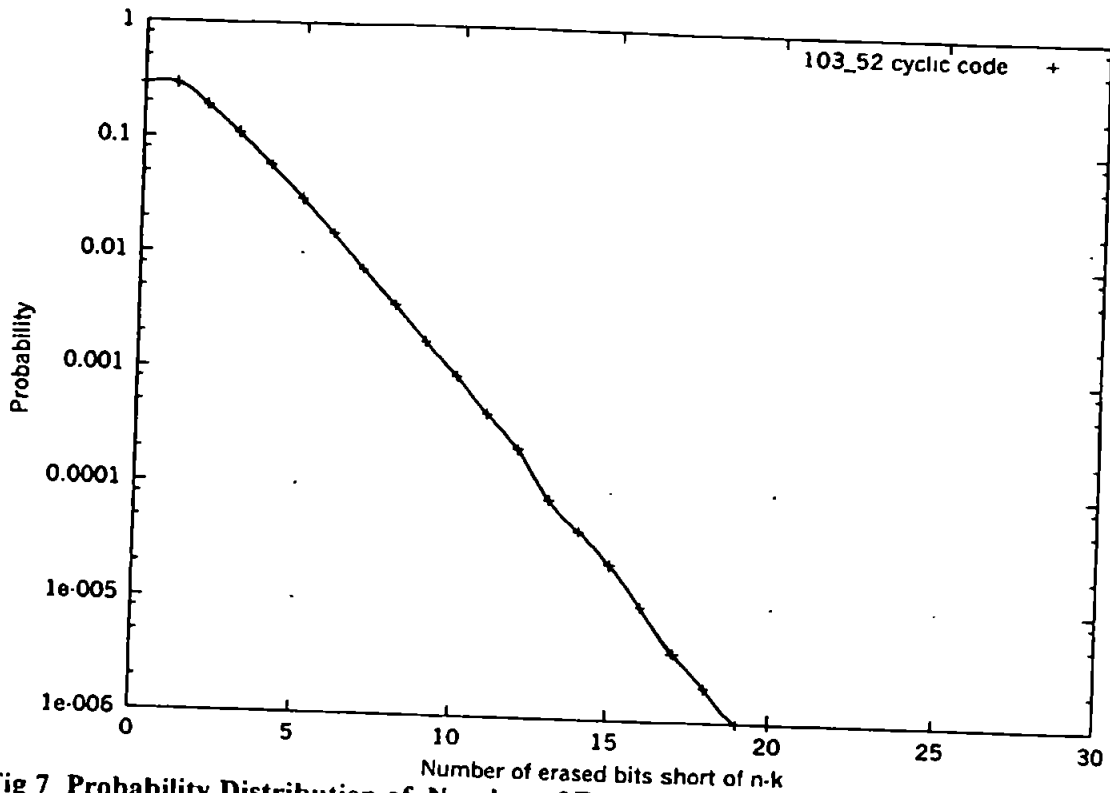


Fig 7 Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctible (n-k) for (103,52) quadratic residue code

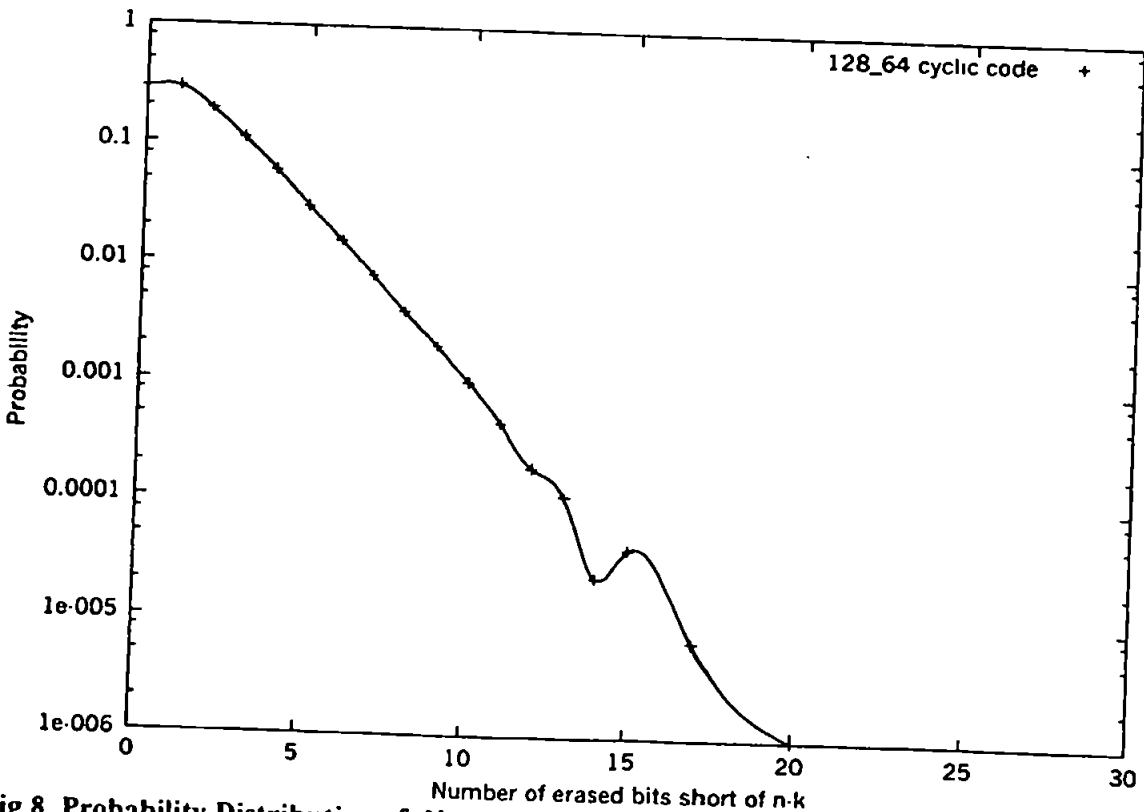


Fig 8 Probability Distribution of Number of Erased Bits not Corrected from Maximum Correctible (n-k) for (128,64) extended BCH code



## A SYSTEM FOR THE CORRECTION OF MISSING OR DELETED SYMBOLS

This invention relates to the encoding and decoding of symbols in symbol sequences in order to correct or compensate for deleted or missing symbols. All of the symbol information is retrieved from each symbol sequence despite some of the symbols being missing or deleted.

### BACKGROUND

In many communication systems, storage systems and broadcasting systems information is encoded as a sequence of symbols from a finite alphabet and binary symbols are the most common. This invention is applicable to any finite alphabet and may be used for binary symbols but is not limited to these. In several applications the symbol stream is subject to missing symbols or deleted symbols. Without loss of generality the total number of symbols is denoted by  $n$  and the number of information symbols contained in the symbol stream is denoted by  $k$ . The number of erased symbols is denoted by  $m$ . In communication systems these  $m$  symbols are referred to as erased symbols [1]. In broadcasting systems or in multicasting systems, symbols are transmitted serially. Each symbol could be a data packet of a suitable length in bits. In broadcasting systems or multicasting, at any point in time, only  $r$  symbols may have been received either because of network congestion or interference or because only  $r$  symbols have been transmitted up until now. The system described below enables the  $k$  information symbols to be retrieved with a given probability of symbol error, provided  $r$  is greater than or equal to  $k$ . In information storage systems, such as hard drives, the invention described below may be used to recover the stored information before all of the  $n$  symbols of data have been read (so as to achieve faster access) or to compensate for unread /missing symbols. It is shown that the system can cope with a much higher number of missing symbols than the current state of the art, for the same transmitted sequence. For example if 64 information bits are encoded into 128 bits by using a code, such as the extended (128,64) BCH code given in the Appendix, then the traditional hard decision decoder [2] will be able to correct only up to 21 missing bits, one bit less than the  $d_{min}$  of the code which is 22. With the invention described below, up to 62 missing bits on average can be corrected and 45 missing bits can be corrected with a probability of 0.99999.

### DESCRIPTION OF THE INVENTION

The information to be stored or transmitted consists of  $k$  information symbols or packets, which are encoded into  $n$  code symbols by using a series of parity check equations. This is a well known procedure described generally, for example, in references [1] and [2]. The  $n$  code symbols are transmitted or stored. A schematic diagram for the encoder is shown in Fig 1. The invention provides a means of recovering the  $k$  information symbols or packets from  $k+r$  symbols, where  $r$  is a small integer between 0 and some limit,  $r_{max}$ . The reasons for wanting to recover the  $k$  information symbols or packets from  $k+r$  symbols are manifold. Some of the remaining symbols may have been corrupted in storage, and are missing, or in a communication system, congestion, time constraints, distortion, noise or other transmission impairments may have prevented the reception of the remaining symbols or packets.

The code that is used for encoding is not the subject of this invention. The open literature is a rich source of suitable codes using binary or non binary symbols and having various values of  $k$  and  $n$ . See for example [1] and [2].

The invention uses the received symbols and the parity check equations in three methods, Method A, Method B, and Method C, each with increasing levels of complexity of the decoder so that in the implementation of the invention, performance may be traded off against cost.

#### Method A

In method A the decoder inserts the received symbols into the parity check equations substituting symbols representing unknown variables for the missing symbols. The equations are scanned for the number of unknowns in each equation. Those equations with only one unknown symbol, are solved for the unknown symbol and the solved symbols are substituted back into the parity check equations. The procedure then repeats and continues repeating until all missing symbols have been solved and found. A schematic diagram for the decoder is shown in Fig 2. Without loss of generality the invention is described further by way of example using binary symbols and with a short code length of 15.

Consider the parity check matrix for a (15,7) error correcting code with a  $d_{min}$  of 5. The code is guaranteed to correct 4 erasures using syndrome decoding [2]. By using the decoding Method A, the average number of erasures corrected is approximately 5.5. The code length is 15 and there are 7 information bits encoded into a total of 15 bits. The sequence to be transmitted or stored is represented as

$$C(x) = c_0 + c_1x^{-1} + c_2x^{-2} + c_3x^{-3} + c_4x^{-4} + c_5x^{-5} + c_6x^{-6} + c_7x^{-7} + c_8x^{-8} + c_9x^{-9} + c_{10}x^{-10} + c_{11}x^{-11} + c_{12}x^{-12} + c_{13}x^{-13} + c_{14}x^{-14}$$

The information is contained in the 15 binary coefficients  $c_0, c_1, c_2, c_3, c_4, \dots, c_{14}$  following the encoding according to the parity check equations

There are 8 parity check equations which define 8 of the coefficients  $c_i$ ,

$$c_0 + c_1 + c_3 + c_7 = 0 \quad (1)$$

$$c_1 + c_2 + c_4 + c_8 = 0 \quad (2)$$

$$c_2 + c_3 + c_5 + c_9 = 0 \quad (3)$$

$$c_3 + c_4 + c_6 + c_{10} = 0 \quad (4)$$

$$c_4 + c_5 + c_7 + c_{11} = 0 \quad (5)$$

$$c_5 + c_6 + c_8 + c_{12} = 0 \quad (6)$$

$$c_6 + c_7 + c_9 + c_{13} = 0 \quad (7)$$

$$c_7 + c_8 + c_{10} + c_{14} = 0 \quad (8)$$

The 7 information bits may be distributed amongst the 15 coefficients in several ways, the usual convention is that the coefficients  $c_0$  through to  $c_6$  are set equal to the information bits and  $c_7$  through to  $c_{14}$  are parity check bits derived from  $c_0$  through to  $c_6$ .

As an alternative representation, the parity check equations may be represented as a parity check matrix:

```

110100010000000
011010001000000
001101000100000
000110100010000
000011010001000
000001101000100
000000110100010
000000011010001

```

where in each row of the matrix the position of the 1's indicate the coefficients used in the parity check equation corresponding to that row

Consider a sequence of missing bits or erasures in positions 3, 12, 1, 9, 2, 0  
The received or the available sequence is therefore represented as

$$c_4x^4 + c_5x^5 + c_6x^6 + c_7x^7 + c_8x^8 + c_{10}x^{10} + c_{11}x^{11} + c_{13}x^{13} + c_{14}x^{14}$$

The 6 coefficients  $c_0, c_1, c_2, c_3, c_9,$  and  $c_{12}$  are unknown and need to be determined unambiguously. This is impossible with the conventional syndrome decoder as only 4 coefficients can be determined as the code has a  $d_{min}$  of 5.

For each erasure, the erasure is represented as an unknown in each of the parity check equations that it appears as  $z_i$ , where  $i$  is the position of the erasure

The set of parity check equations become

$$z_0 + z_1 + z_3 + c_7 = 0 \quad (A1)$$

$$z_1 + z_2 + c_4 + c_8 = 0 \quad (A2)$$

$$z_2 + z_3 + c_5 + z_9 = 0 \quad (A3)$$

$$z_3 + c_4 + c_6 + c_{10} = 0 \quad (A4)$$

$$c_4 + c_5 + c_7 + c_{11} = 0 \quad (A5)$$

$$c_5 + c_6 + c_8 + z_{12} = 0 \quad (A6)$$

$$c_6 + c_7 + z_9 + c_{13} = 0 \quad (A7)$$

$$c_7 + c_8 + c_{10} + c_{14} = 0 \quad (A8)$$

All the equations are scanned to determine the number of unknowns  $z_i$  in each equation. Those equations with only one unknown are solved. These are equations (A4), (A6) and (A7) and  $z_3, z_9$  and  $z_{12}$  are solved to produce  $c_3, c_9$  and  $c_{12}$ . That is

$$c_3 = c_4 + c_6 + c_{10}$$

$$c_9 = c_6 + c_7 + c_{13} \text{ and}$$

$$c_{12} = c_5 + c_6 + c_8$$

These are then substituted into equations (A1) through to (A8) to produce the following equations

$$z_0 + z_1 + c_3 + c_7 = 0 \quad (B1)$$

$$z_1 + z_2 + c_4 + c_8 = 0 \quad (B2)$$

$$z_2 + c_3 + c_5 + c_9 = 0 \quad (B3)$$

$$c_3 + c_4 + c_6 + c_{10} = 0 \quad (B4)$$

$$c_4 + c_5 + c_7 + c_{11} = 0 \quad (B5)$$

$$c_5 + c_6 + c_8 + c_{12} = 0 \quad (B6)$$

$$c_6 + c_7 + c_9 + c_{13} = 0 \quad (B7)$$

$$c_7 + c_8 + c_{10} + c_{14} = 0 \quad (B8)$$

The procedure then repeats, scanning the equations to determine the number of unknowns  $z_i$  in each equation. Those equations with only one unknown are solved. In this case it is equation (B3) for the unknown  $z_2$  in order to find  $c_2$ . The solution (s) is substituted to produce a new set of equations

$$\begin{aligned} z_0 + z_1 + c_3 + c_7 &= 0 & (C1) \\ z_1 + c_2 + c_4 + c_8 &= 0 & (C2) \\ c_2 + c_3 + c_5 + c_9 &= 0 & (C3) \\ c_3 + c_4 + c_6 + c_{10} &= 0 & (C4) \\ c_4 + c_5 + c_7 + c_{11} &= 0 & (C5) \\ c_5 + c_6 + c_8 + c_{12} &= 0 & (C6) \\ c_6 + c_7 + c_9 + c_{13} &= 0 & (C7) \\ c_7 + c_8 + c_{10} + c_{14} &= 0 & (C8) \end{aligned}$$

The procedure then repeats, scanning the equations to determine the number of unknowns  $z_i$  in each equation. Those equations with only one unknown are solved which is now equation (C2) for the unknown  $z_1$  in order to find  $c_1$ . The solution (s) is substituted in the equations to produce a new set of equations of which there is only one unknown  $z_0$  and the new equation (D0) solved to find  $c_0$ . In this way all of the erasures have been corrected. As shown in Fig 2 the received codeword is clocked into the n stage, tri-state, shift register, with each stage i storing one of 3 states, either 0 or 1 or  $z_i$ , to represent an erasure in that position. Each shift register stage feeds the set of parity check equations and those with only one  $z$  entry are solved and the solutions fed back to the respective shift register stage as shown in Fig 2. The procedure then repeats until the shift register contains no  $z$  states or until decoding fails due to an excessive number of erasures being present in the parity check equations.

#### Method B

In the event that each parity check equation contains two or more erasures then the procedure of Method A will fail. Method B is the same as Method A except that in the event of all parity check equations containing two or more erasures, one or more erased bits are selected and their states are systematically guessed. The basis of the selection, is that these erased bits are in the maximum number of equations containing only two erasures. The states of these selected bits are set to all possible symbol states, one at a time, substituted back into the parity check equations, and Method A invoked. A schematic diagram of the decoder is shown in Fig 3. If the procedure progresses with all the parity check equations solved then decoding is declared complete. In the event that all of the parity check equations cannot be solved, then the received symbols are input again from the receive buffer, and new guesses are made for the selected bits. If all possible guesses have been made for the selected bits, then a new selection of bits to be guessed is made and the procedure repeated.

#### Method C

For the ultimate performance all of the information contained in the parity check equations needs to be used with a consequent increase in decoder complexity. The codeword is received and unknowns  $z_i$  substituted in positions of erased symbols in the parity check equations. Starting with one of the erased symbols,  $z_n$ , the first equation containing this symbol is flagged that it will be used for the solution of  $z_n$  and then this equation is subtracted from all other equations containing  $z_n$  and not yet flagged, to produce a new set of equations. The procedure repeats with the next of the non flagged equations containing the next erased symbol  $z_{n-1}$  flagged and subtracted from all of the remaining non flagged equations containing  $z_{n-1}$ . The procedure is a form of Gaussian reduction of the parity check equations.

The procedure repeats until either no non flagged equations remain containing the erased symbol  $z_{n-1}$  (in which case a decoder failure is declared) or no erased symbols remain that are not in flagged equations. In this case starting with the last flagged equation with erased symbol  $z_{last}$  this equation is solved to find  $c_{last}$  and this equation is unflagged. This coefficient is substituted back into the remaining flagged equations containing  $z_{last}$ . The procedure now repeats with the second from last flagged equation now being solved for  $z_{last-1}$ : this equation is unflagged and followed by back substitution of  $c_{last-1}$  for  $z_{last-1}$  in the remaining flagged equations. A block schematic of the decoder is shown in Fig 4. The received symbols are stored in the shift register with the erased symbols being replaced by the unknowns  $z_i$ . The Gaussian reduced equations are computed and used to define the connection of symbol adders from each respective shift register stage to compute the outputs  $d_1$  through to  $d_n$ . The non erased symbols contained in the shift register are switched directly through to their respective outputs so that overall, the decoded codeword containing no erased symbols is present at the outputs  $d_1$  through to  $d_n$ .

As an example of the method consider the (15,7) code with erasures in positions  $c_1, c_2, c_3, c_4$ . After substitution with the unknowns in the parity check equations, the following set of equations are obtained:

$$\begin{aligned} c_0 + z_1 + z_3 + c_7 &= 0 & (D1) \\ z_1 + z_2 + z_4 + c_8 &= 0 & (D2) \\ z_2 + z_3 + c_5 + c_9 &= 0 & (D3) \\ z_3 + z_4 + c_6 + c_{10} &= 0 & (D4) \\ z_4 + c_5 + c_7 + c_{11} &= 0 & (D5) \\ c_5 + c_6 + c_8 + c_{12} &= 0 & (D6) \\ c_6 + c_7 + c_9 + c_{13} &= 0 & (D7) \\ c_7 + c_8 + c_{10} + c_{14} &= 0 & (D8) \end{aligned}$$



There are a total of  $(s+1)k$  information symbols which are encoded using the parity check equations of a selected code into a total number of transmitted symbols equal to  $(s+1)n$ . The symbols are transmitted in a series of packets with each packet corresponding to a row of the matrix as indicated above. For example the row

$$b_{20} \ b_{21} \ b_{22} \ b_{23} \ b_{24} \ b_{25} \ b_{26} \ b_{27} \quad b_{2s}$$

is transmitted as a single packet.

Self contained codewords are encoded from each column of  $k$  symbols. For example  $b_{00} \ b_{10} \ b_{20} \ b_{30} \ b_{40} \quad b_{k-10}$  form the  $k$  information symbols of one codeword and the remaining symbols  $b_{10} \ b_{k+10} \ b_{k+20} \quad b_{n-10}$  are the  $n-k$  parity symbols of that codeword and these are the result of encoding the  $k$  information symbols.

As a result of network congestion, drop outs, loss of radio links or other multifarious reasons, not all of the transmitted packets are received. The effect is that some rows above may be considered as erased rows. The decoding procedure is that codewords are assembled from the received packets with missing symbols corresponding to the missing packets marked as  $z_j$  corresponding to their position in the matrix. For example if the second packet only is missing above

The first received codeword corresponds to the first column above and is

$$b_{00} \ z_{10} \ b_{20} \ b_{30} \ b_{40} \ b_{50} \ b_{60} \ b_{70} \quad b_{n-10}$$

The second codeword corresponding to the second column above is

$$b_{01} \ z_{11} \ b_{21} \ b_{31} \ b_{41} \ b_{51} \ b_{61} \ b_{71} \quad b_{n-11}$$

and so on

All of the three methods outlined above may be used to solve for the erased symbol  $z_{10}$  in the first received codeword, and for the erased symbol  $z_{11}$  in the second received codeword and so on up to the  $s$ 'th codeword (column) solving for symbol  $z_{1s-1}$ . As an example the binary, extended (128,64) BCH code given in the Appendix could be used to encode the information data. The packet length is chosen to be 100 bits, and the total transmission could consist of 128 transmitted packets (12,800 bits total) containing 6,400 bits of information. On average as soon as any 66 packets from the original 128 packets have been received, the remaining 62 packets are treated as if they are erased. The 100 codewords are assembled, and decoded with the results that the erased symbols are solved and the 6,400 bits of information retrieved. One additional advantage is that a user does not have to wait until the entire transmission has been received in order to recover the 6,400 bits of information even if there have been no erasures. For this code, on average, only 66 packets have to be received to recover all 6,400 bits of information. (see results below for this code's performance)

### Results for Some Typical Codes

The applicability of the decoding methods above depends upon the error correcting code being used and specifically on the parity check matrix being used. The parity check matrix should be sparse (each row of the matrix having a small number of non zero entries) for Methods A and B. The sparseness of the parity check matrix does not affect the performance of Method C. A particularly strong binary code and one which has a sparse parity check matrix is the (255,175) binary code given in the Appendix. This code has a length of 255 bits after encoding of 175 information bits. The performance of this code for the three methods above is shown in Fig 5 in terms of the probability of decoder error (FER) as a function of the erasure probability for every transmitted bit. An erasure probability of 0.2 means that on average 1 bit in 5 is erased or lost. Method C has the best performance but at the expense of decoder complexity. The ultimate performance of this method as a function of error correcting code is shown in Fig 6 for the example (255,175) code which can correct a maximum of 80 erased bits. Fig 6 shows the probability density function of the number of erased bits short of the maximum correctible which is  $n-k$ . The results were obtained by computer simulations. The probability of being able to correct only 68 bits, a shortfall of 12 bits, is  $1.1 \times 10^{-3}$ . Simulations indicate that on average 77.6 erased bits may be corrected for this code. In comparison the BCH (255,178) code having similar rate is also shown in Fig 6. The BCH code has similar a similar rate but a higher minimum Hamming distance of 22 (compared to 17). It can be seen that it has better performance than the (255,175) code but it has a less sparse parity check matrix and consequently it is less suitable for the decoding Methods A and B. Moreover the average shortfall in erasures not being able to be corrected is virtually identical for the two codes.

The simulation results of using Method C for the (103,52) quadratic residue binary code [3] are shown in Fig 7. The minimum Hamming distance for this code is 19 and the results are similar to that of the (255,178) BCH code above. It is found from the simulations that on average 49.1 erasure bits are corrected (out of a maximum of 51) and the average shortfall from the maximum is 1.59 bits.

Similarly the results for the extended BCH (128,64) code are shown in Fig 8. This code has a minimum Hamming distance of 22 and has a similar probability density function to the other BCH code above. On average 62.39 erasure bits are corrected (out of a maximum of 64) and the average shortfall is 1.61 bits from the maximum.

### References

- [1] W.W.Peterson, Error Correcting Codes, The MIT Press 1961
- [2] S.Lin and D.J. Costello, Error Control Coding, Fundamentals and Applications, Prentice-Hall 1983

[3] F.J. MacWilliams and N.J.A. Sloane, The Theory of Error Correcting Codes, North-Holland 1977.

### Appendix

The parity check (H) matrix below is for the (255,175) binary code having a minimum Hamming distance of 17. It is a sparse matrix because a code of this length and performance would usually have approximately 80 entries per row instead of 16. Consequently it is particularly suitable for use in Method A and Method B. The -1 symbol at the end of each row is only there to enable the matrix to be easily machine readable, it is not part of the code. The numbers represent the bit positions of the bits involved in each parity check equation.

```

0 18 26 32 56 61 97 106 110 113 125 150 152 172 173 183 -1
1 19 27 33 57 62 98 107 111 114 126 151 153 173 174 184 -1
2 20 28 34 58 63 99 108 112 115 127 152 154 174 175 185 -1
3 21 29 35 59 64 100 109 113 116 128 153 155 175 176 186 -1
4 22 30 36 60 65 101 110 114 117 129 154 156 176 177 187 -1
5 23 31 37 61 66 102 111 115 118 130 155 157 177 178 188 -1
6 24 32 38 62 67 103 112 116 119 131 156 158 178 179 189 -1
7 25 33 39 63 68 104 113 117 120 132 157 159 179 180 190 -1
8 26 34 40 64 69 105 114 118 121 133 158 160 180 181 191 -1
9 27 35 41 65 70 106 115 119 122 134 159 161 181 182 192 -1
10 28 36 42 66 71 107 116 120 123 135 160 162 182 183 193 -1
11 29 37 43 67 72 108 117 121 124 136 161 163 183 184 194 -1
12 30 38 44 68 73 109 118 122 125 137 162 164 184 185 195 -1
13 31 39 45 69 74 110 119 123 126 138 163 165 185 186 196 -1
14 32 40 46 70 75 111 120 124 127 139 164 166 186 187 197 -1
15 33 41 47 71 76 112 121 125 128 140 165 167 187 188 198 -1
16 34 42 48 72 77 113 122 126 129 141 166 168 188 189 199 -1
17 35 43 49 73 78 114 123 127 130 142 167 169 189 190 200 -1
18 36 44 50 74 79 115 124 128 131 143 168 170 190 191 201 -1
19 37 45 51 75 80 116 125 129 132 144 169 171 191 192 202 -1
20 38 46 52 76 81 117 126 130 133 145 170 172 192 193 203 -1
21 39 47 53 77 82 118 127 131 134 146 171 173 193 194 204 -1
22 40 48 54 78 83 119 128 132 135 147 172 174 194 195 205 -1
23 41 49 55 79 84 120 129 133 136 148 173 175 195 196 206 -1
24 42 50 56 80 85 121 130 134 137 149 174 176 196 197 207 -1
25 43 51 57 81 86 122 131 135 138 150 175 177 197 198 208 -1
26 44 52 58 82 87 123 132 136 139 151 176 178 198 199 209 -1
27 45 53 59 83 88 124 133 137 140 152 177 179 199 200 210 -1
28 46 54 60 84 89 125 134 138 141 153 178 180 200 201 211 -1
29 47 55 61 85 90 126 135 139 142 154 179 181 201 202 212 -1
30 48 56 62 86 91 127 136 140 143 155 180 182 202 203 213 -1
31 49 57 63 87 92 128 137 141 144 156 181 183 203 204 214 -1
32 50 58 64 88 93 129 138 142 145 157 182 184 204 205 215 -1
33 51 59 65 89 94 130 139 143 146 158 183 185 205 206 216 -1
34 52 60 66 90 95 131 140 144 147 159 184 186 206 207 217 -1
35 53 61 67 91 96 132 141 145 148 160 185 187 207 208 218 -1
36 54 62 68 92 97 133 142 146 149 161 186 188 208 209 219 -1
37 55 63 69 93 98 134 143 147 150 162 187 189 209 210 220 -1
38 56 64 70 94 99 135 144 148 151 163 188 190 210 211 221 -1
39 57 65 71 95 100 136 145 149 152 164 189 191 211 212 222 -1
40 58 66 72 96 101 137 146 150 153 165 190 192 212 213 223 -1
41 59 67 73 97 102 138 147 151 154 166 191 193 213 214 224 -1
42 60 68 74 98 103 139 148 152 155 167 192 194 214 215 225 -1
43 61 69 75 99 104 140 149 153 156 168 193 195 215 216 226 -1
44 62 70 76 100 105 141 150 154 157 169 194 196 216 217 227 -1
45 63 71 77 101 106 142 151 155 158 170 195 197 217 218 228 -1
46 64 72 78 102 107 143 152 156 159 171 196 198 218 219 229 -1
47 65 73 79 103 108 144 153 157 160 172 197 199 219 220 230 -1
48 66 74 80 104 109 145 154 158 161 173 198 200 220 221 231 -1
49 67 75 81 105 110 146 155 159 162 174 199 201 221 222 232 -1
50 68 76 82 106 111 147 156 160 163 175 200 202 222 223 233 -1
51 69 77 83 107 112 148 157 161 164 176 201 203 223 224 234 -1
52 70 78 84 108 113 149 158 162 165 177 202 204 224 225 235 -1
53 71 79 85 109 114 150 159 163 166 178 203 205 225 226 236 -1
54 72 80 86 110 115 151 160 164 167 179 204 206 226 227 237 -1
55 73 81 87 111 116 152 161 165 168 180 205 207 227 228 238 -1
56 74 82 88 112 117 153 162 166 169 181 206 208 228 229 239 -1
57 75 83 89 113 118 154 163 167 170 182 207 209 229 230 240 -1
58 76 84 90 114 119 155 164 168 171 183 208 210 230 231 241 -1
59 77 85 91 115 120 156 165 169 172 184 209 211 231 232 242 -1
60 78 86 92 116 121 157 166 170 173 185 210 212 232 233 243 -1
61 79 87 93 117 122 158 167 171 174 186 211 213 233 234 244 -1
62 80 88 94 118 123 159 168 172 175 187 212 214 234 235 245 -1
63 81 89 95 119 124 160 169 173 176 188 213 215 235 236 246 -1
64 82 90 96 120 125 161 170 174 177 189 214 216 236 237 247 -1
65 83 91 97 121 126 162 171 175 178 190 215 217 237 238 248 -1
66 84 92 98 122 127 163 172 176 179 191 216 218 238 239 249 -1

```



56 58 60 61 62 64 65 69 70 72 73 74 76 80 81 83 85 86 90 91 92 93 94 95 96 99 100 108 109 113 114 116 118 120-1  
 57 59 61 62 63 65 66 70 71 73 74 75 77 81 82 84 86 87 91 92 93 94 95 96 97 100 101 109 110 114 115 117 119 121-1  
 58 60 62 63 64 66 67 71 72 74 75 76 78 82 83 85 87 88 92 93 94 95 96 97 98 101 102 110 111 115 116 118 120 122-1  
 59 61 63 64 65 67 68 72 73 75 76 77 79 83 84 86 88 89 93 94 95 96 97 98 99 102 103 111 112 116 117 119 121 123-1  
 60 62 64 65 66 68 69 73 74 76 77 78 80 84 85 :7 89 90 94 95 96 97 98 99 100 103 104 112 113 117 118 120 122 124-1  
 61 63 65 66 67 69 70 74 75 77 78 79 81 85 86 88 90 91 95 96 97 98 99 100 101 104 105 113 114 118 119 121 123 125-1  
 62 64 66 67 68 70 71 75 76 78 79 80 82 86 87 89 91 92 96 97 98 99 100 101 102 105 106 114 115 119 120 122 124 126-1  
 0 63 65 67 68 69 71 72 76 77 79 80 81 83 87 88 90 92 93 97 98 99 100 101 102 103 106 107 115 116 120 121 123 125-1

As before the notation is that each row contains the positions of bits in that equation. There are 64 rows because there are 64 equations. The k information bits (also 64) may be in any position but traditionally these are in positions 0 to 63.



## CLAIMS

Claim 1. A system in which  $k$  information symbols are encoded into  $n$  symbols using parity check equations from an error correcting code and some of the  $n$  symbols are marked as erased symbols. The  $k$  information symbols are retrieved from the remaining symbols based on a decoder (Method A above) which examines the number of erasures in each parity check equation and solves for the erased symbols in those parity check equations that only contain one erased symbol. All of these erased symbols are determined and substituted back into the parity check equations and the procedure is repeated over and over again until all erased symbols have been determined and all  $k$  information symbols retrieved.

Claim 2. A system according to Claim 1 and in which the  $n$  symbols containing marked erasure symbols are stored in a buffer memory. Under the condition that all of the parity check equations contain two or more erased symbols then one or more of these symbols are guessed as to their respective states using all combinations of their respective states, one state at a time, and the guesses substituted into the parity check equations as described in Method B. Each parity check equation which has only one erased symbol is solved for that symbol and all the solved symbols substituted into the equations and the procedure repeated as in Claim 1. In the event that not all equations are solved the original  $n$  symbols are retrieved from the buffer memory and the whole procedure repeated with new guesses for one or more of the erased symbols until all parity check equations are solved or a decoder failure is declared.

Claim 3. A system in which  $k$  information symbols are encoded into  $n$  symbols using parity check equations from an error correcting code and in which some of the  $n$  symbols are marked as erased symbols. The  $k$  information symbols are retrieved based on a decoder (Method C above) which selects one erased symbol at a time. The parity check equations are examined and the first equation containing this symbol is flagged that it will be used to find this symbol. Each unflagged equation containing this symbol is replaced with the result of that unflagged equation minus the equation just flagged. The procedure is repeated examining all unflagged equations for the presence of the next selected erased symbol. The first unflagged equation found is flagged and subtracted from all other unflagged equations containing that symbol. The procedure is repeated over and over again until each erased symbol has a corresponding flagged equation and either there are no erased symbols left that have not been selected or there are no unflagged equations containing the currently selected erased symbol. In this latter event a decoder failure is declared. The last flagged equation is used to solve for its respectively selected erased symbol and the solved symbol is substituted into all equations in which it is present. The next to last flagged equation is solved for its selected erased symbol and then the solved symbol is substituted into all equations in which it is present. The procedure is repeated over and over again working through the flagged equations, in last to be flagged order until all erased symbols have been solved.

Claim 4. A system in which the product of  $k$  and  $s$  ( $k \cdot s$ ) information symbols are encoded into  $n \cdot s$  symbols using parity check equations from an error correcting code and transmitted or stored as packets of length  $s$  symbols. The encoding is carried out so that each packet contains a single coordinate symbol from each of  $s$  encoded codewords. With  $k$  or more packets received or recovered the remaining packets are marked as being erased. The symbols within these packets are marked as erased symbols and the corresponding  $s$  codewords each decoded using one of the three Methods A, B or C. If successful decoding is not possible either an additional non erased packet is obtained and the decoding procedure attempted again or a decoding failure is declared.

Claim 5. A system of multicasting or broadcasting in which the information to be transmitted or stored is partitioned into blocks of  $k$  packets of fixed length or of variable length equal to  $s$  symbols and encoded according to Claim 4 into  $n$  packets of length  $s$  symbols. As soon as  $k$  or more packets have been received or recovered the  $k \cdot s$  information symbols corresponding to that partition are decoded using one of the Methods A, B or C. In this way a system is provided in which information may be multicast or broadcast in minimum time and also be resilient to lost packets.

**CLAIMS**

**Amendments to the claims have been filed as follows**

Claim 1. A system in which  $k$  information symbols are encoded into  $n$  symbols using parity check equations from an error correcting code and in which some of the  $n$  symbols are marked as erased symbols with  $k$  information symbols retrieved based on a decoder which selects one erased symbol at a time and flags the first parity check equation containing this symbol and replaces each unflagged equation containing this symbol with that unflagged equation minus the equation just flagged continuing the procedure until each erased symbol has a corresponding flagged equation and then the last flagged equation is used to solve for its respectively selected erased symbol and the solved symbol is substituted into all equations in which it is present followed by the next to last flagged equation which is solved for its selected erased symbol with this solved symbol substituted into all equations in which it is present followed by working through the remaining flagged equations, in last to be flagged order, until all erased symbols have been solved.

Claim 2. A system according to Claim 1, in which the product of  $k$  and  $s+1$ ,  $(k \cdot s+k)$  information symbols are encoded into  $n \cdot s+n$  symbols using parity check equations from an error correcting code and transmitted or stored as packets of length  $s+1$  symbols with the encoding carried out so that each packet contains a single coordinate symbol from each of  $s+1$  encoded codewords and once  $k$  or more packets have been received or recovered the remaining packets are marked as being erased with the symbols contained in these packets marked as erased symbols and the corresponding  $s+1$  codewords each decoded such that the  $ks+k$  information symbols are retrieved.

Claim 3 A system according to Claim 2, for application in multicasting, or broadcasting, in which the information to be transmitted or stored is partitioned into blocks of  $k$  packets of fixed length equal to  $s+1$  symbols and encoded into  $n$  packets of length  $s+1$  symbols and in which as soon as  $k$  or more packets have been received or recovered the  $ks+k$  information symbols are decoded such that the  $ks+k$  information symbols are retrieved





11

Application No: GB0428042.6

Examiner: Dr Mark Shawcross

Claims searched: 1, 4 & 5

Date of search: 8 February 2006

**Patents Act 1977: Search Report under Section 17**

**Documents considered to be relevant:**

Category	Relevant to claims	Identity of document and passage or figure of particular relevance
X	1 & 4-5	US 5115436 A (McAULEY) in particular, see col.3 line 31 to col.4 line 68 and figures 1 & 6
X	1 & 4-5	EP 0903955 A1 (STMicroelectronics) in particular, see paragraphs [0026-0042].
X	1 & 4-5	US 4555784 A (WOOD) col.6 line 28 to col.12 line 41

**Categories:**

X Document indicating lack of novelty or inventive step	A Document indicating technological background and/or state of the art.
Y Document indicating lack of inventive step if combined with one or more other documents of same category.	P Document published on or after the declared priority date but before the filing date of this invention.
& Member of the same patent family	E Patent document published on or after, but with priority date earlier than, the filing date of this application.

**Field of Search:**

Search of GB, EP, WO & US patent documents classified in the following areas of the UKC<sup>x</sup> :

G4A

Worldwide search of patent documents classified in the following areas of the IPC

G06F

The following online and other databases have been used in the preparation of this search report

Online: EPODOC, WPI, INSPEC