Shortening and turbo decoding of product codes

Lim, Kai Ching

2007

Lim, K. C. (2007). Shortening and turbo decoding of product codes. Master's thesis,
Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/46902>

<https://doi.org/10.32657/10356/46902>

Nanyang Technological University

Downloaded on 26 Apr 2023 17:30:48 SGT

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**SHORTENING AND TURBO DECODING
OF
PRODUCT CODES**

**LIM KAI CHING
SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY
2007**

Shortening and Turbo Decoding Of Product Codes

Lim Kai Ching

School of Electrical & Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfillment of the requirement for the degree of
Master of Engineering

2007

QA
268
L732
2007

Acknowledgement

I would like to express my heart-felt gratitude and thanks to my supervisor, Associate Professor Guan Yong Liang for his guidance, advice and patience. His help has very much steered the directions of my work and made my work possible. I greatly thank and appreciate his efforts. I would like to dedicate the following poem to him.

Teachers
Paint their minds
and guide their thoughts
Share their achievements
and advise their fault

Inspire a Love
of knowledge and truth
As you light the path
Which leads our youth

For our future brightens
with each lesson you teach
Each smile you lengthen
Each goal you help reach

For the dawn of each poet
each philosopher and king
Begins with a Teacher
And the wisdom they bring

Kevin William Huff

Lastly but not the least, I would like to thank Associate Professor Law Choi Look for his kind support during my stay in Positioning and Wireless Technology Centre (PWTC).

Summary

Product codes (PC) have become an important class of error correction codes in recent years. They have been adopted in the IEEE 802.16 standard for wireless metropolitan area networks (WMAN) and received wide industry acceptance as a good forward error correction (FEC) solution. They exhibit near-capacity bit-error-rate (BER) performance and reasonable decoding complexity with soft decoding. Moreover, unlike convolutional-based Turbo codes, product codes suffer from lower error floors.

In communication systems involving block and product codes, code shortening of a code is commonly used to flexibly adjust the code length or code rate to accommodate protocol. In this research, we first look into the BER performance analysis of shortened and shortened-extended product codes. Our achievement is the derivation of useful code weight enumerator expressions based on the widely known properties of the original code (before extension or shortening) and the number of nullified information symbols. They have been verified to be tight for Hamming, BCH and Golay codes. Such weight enumerator expressions lead naturally to useful decoding BER performance bounds and coding loss formulas.

Secondly, we found that the existing code shortening approach specified in the IEEE 802.16a WiMax standard is not optimal in terms of minimum Hamming distance. Hence we set out to find better code shortening scheme. The result is a novel "*Ordered Shortening*" scheme which produces much improved code distance as well as code weight enumerator distributions. These improved code properties lead to improved BER performance for the *ordered-shortened* codes, as verified by computer simulations using

the Pyndiah decoding algorithm. Furthermore, the proposed *ordered shortening* scheme is also shown to be able to meet the theoretical upper bound values of minimum Hamming distance for some codes.

Finally, we have investigated the turbo decoding performance of product codes in pulsed or partial-band jamming channels, which are of particular interest to military/defense communication systems. We have developed the BER bounds of turbo product codes in pulsed jamming channel with Rayleigh fading, and optimised the Pyndiah-Chase algorithm for such channels. The results show that with the use of jamming state information and channel state information, the simulated BERs of the product codes are able to approach the asymptotic BER bounds.

Table of Content

Acknowledgement..... i

Summary.....ii

Conventions of Symbols Notationsxii

Chapter 1 Introduction 1

1.1 Literature Review and Motivation 1

1.2 Objectives..... 4

1.3 Major Contributions of the Thesis 5

1.4 Organization of the Thesis..... 6

Chapter 2 Background 8

2.1 Linear Block Codes 8

2.1.1 Linear Block Codes 8

2.1.2 Matrix Description of Linear Block Codes 8

2.1.3 Minimum Hamming distance 10

2.1.4 Weight Enumerator 10

2.1.5 Families of Linear Block Codes Considered in this Thesis..... 11

2.1.5.1 Hamming Codes 11

2.1.5.2 BCH Codes..... 12

2.2 Product Codes..... 12

2.2.1 Product Codes 12

2.2.2 Weight Enumerator of Product codes..... 14

2.3 Code Shortening/Extension..... 15

Table of Content	v
2.3.1 Extended Code	15
2.3.2 Shortened Code.....	16
2.4 Decoding BER Union Bounds	17
2.4.1 Decoding BER Union Bounds in AWGN Channel.....	17
2.4.2 Decoding BER Union Bounds in Rayleigh Fading Channel	17
2.5 Pyndiah's SISO Decoding Algorithm of Product Codes for AWGN Channel	19
2.5.1 Soft Decoding of Linear Block Code.....	20
2.5.1.1 AWGN Channel Model	20
2.5.1.2 Maximum Likelihood Decoding	21
2.5.1.3 Chase's Selection of Codewords.....	22
2.5.2 Soft Decoding Output of Linear Block Codes	23
2.5.2.1 Soft Output - Reliability of Decision d	23
2.5.2.2 Extrinsic Information	26
2.5.3 Turbo Decoding of Product Codes	27
2.6 Pyndiah's SISO Decoding Algorithm of Product Code for Rayleigh Fading Channel	29
2.6.1 Rayleigh Flat Fading Channel Model	29
2.6.2 Decoding with Channel State Information	30
2.6.3 Decoding Without Channel State Information.....	32
Chapter 3 Effect of Code Shortening on BER Performance	33
3.1 Weight Enumerator of Shortened Code.....	33
3.1.1 Approximation to the Weight Distribution of Shortened Binary Linear Codes.....	33
3.1.2 Comparison of True and Approximated weight enumerator of Shortened Code	39
3.2 Weight Enumerator of Punctured Code	43

3.2.1 Distance Distribution of Punctured Codes.....	44
3.2.2 Approximated Weight Enumerator for Punctured Code with Single Punctured Parity Bit	44
3.2.3 Approximated Weight Enumerator with Numerous Punctured Parity Bits.....	47
3.2.4 Comparison of True and Approximated Weight Enumerators of Punctured Codes	47
3.3 BER Bounds of Shortened Codes in BER Bounds of Shortened Codes in AWGN Channel	51
3.3.1 BER Bound of Shortened Block Code in AWGN Channel	51
3.3.2 BER Bound for Shortened Product code in AWGN Channel.....	51
3.3.3 BER Bound for Extended-Shortened Codes	52
3.4 BER Bounds of Shortened Codes in Rayleigh Fading Channel	53
3.5 Asymptotic Coding Loss of Shortened Codes	54
3.6 Simulations and Discussions.....	56
3.6.1 Results for BER Bounds	56
Chapter 4 Ordered Shortening	60
4.1 Motivation	60
4.2 Ordered Shortening	61
4.2.1 Shortening from the Parity Check Matrix Point of View	61
4.2.2 Principle of <i>Ordered Shortening</i>	62
4.3 Algorithm	64
4.3.1 Example of <i>Ordered Shortening</i>	65
4.4 Results	68
4.4.1 <i>Ordered-Shortened</i> Hamming and BCH Codes.....	68

Table of Content	vii
4.4.2 Comparison of Minimum Hamming Distance with Theoretical Upper Bound	70
4.4.3 Comparison of Minimum Weight Enumerator	71
4.4.4 Comparison of Simulated BER	73
Chapter 5 Turbo Decoding of Product Code in Pulsed Jamming Channels	77
5.1 Pulsed Jamming Noise Model	77
5.2 BER Bounds for Pulsed Jamming Channels	78
5.2.1 Pulsed Jamming AWGN Channel with JSI	78
5.2.2 Pulsed Jamming Rayleigh Fading Channel with JSI and CSI	80
5.3 Pyndiah's SISO Decoder for Pulsed Jamming Channel	82
5.3.1 Decoding in Pulsed Jamming AWGN Channel	82
5.3.1.1 Case 1: Decoding with JSI	82
5.3.1.2 Case 2: Decoding without JSI	88
5.2.3 Decoding in Pulsed Jamming Rayleigh Fading Channel	88
5.3.1.3 Case 1: Decoding with JSI and CSI	88
5.3.1.4 Case 2: Decoding without JSI and CSI	89
5.4 Results and Discussions	90
Chapter 6 Conclusions and Recommendations	99
6.1 Conclusions	99
6.2 Recommendations	100
List of Publications	102
Bibliography	103
Appendix A - Verification of Derived Weight Spectrum for Shortened Hamming(31, 26)	108
Appendix B – Verification of Derived Weight Spectrum for Shortened BCH(63, 18)	110

Table of Content	viii
<hr/>	
Appendix C – Derivation of (5.10)	112
Appendix D – Derivation of (5.21)	114

List of Figures

Fig. 2.1. Structure of Systematic Codeword	9
Fig. 2.2. Encoder of a Product Code C^P	13
Fig. 2.3. Structure of Product Code C^P	13
Fig. 2.4. AWGN Channel Model	21
Fig. 2.5. Block diagram of component decoder in Pyndiah's decoder	28
Fig. 2.6. Rayleigh Flat Fading Channel Model.....	29
Fig. 3.1. Comparison of Approximated A_d and true A_d of s-Hamming(6, 3) with $s=1$	35
Fig. 3.2. Comparison of Approximated A_d and true A_d of s-BCH(28,13) with $s=3$	36
Fig. 3.3. Comparison of Approximated A_d and true A_d of s-Hamming(25,20) with $s=6$	36
Fig. 3.4. True distance and weight distributions of punctured codes from Hamming(15,11,3) with $p=1,2$ and 3	45
Fig. 3.5. True distance and weight distributions of punctured-Hamming(8,7) with $p=7$	46
Fig. 3.6. Proposed BER bounds and simulated BER values for Hamming(7, 4) and BCH(15, 7) shortened by 1 bit ($s=1$) in AWGN channel	57
Fig. 3.7. Proposed BER bounds and simulated BER values for BCH(31,16) shortened by $s=8$ and $s=10$ in AWGN channel	58
Fig. 3.8. Proposed asymptotic BER bounds and simulated BER values for s-e-PC(11,6,4) ² shortened by $s^h=s^v=5$ in AWGN and Rayleigh fading channels.....	59
Fig. 4.1. Correspondence between m_i and p_i	62

Fig. 4.2. Comparison of d_{min} between conventional shortening, <i>ordered</i> shortening and the best (31-s, 26-s) possible code. Original code= Hamming(31,26,3).	70
Fig. 4.3. Comparison of A_d between <i>ordered</i> and conventional shortening for (a) s-Hamming(18, 13) and (b) s-PC(18, 13) ²	72
Fig. 4.4. Simulated and analytical BER performance of s-PC(17, 12) ² with <i>ordered</i> and conventional shortening in AWGN channel	74
Fig. 4.5. Simulated and analytical BER performance of s-PC(33, 27) ² with <i>ordered</i> and conventional shortening in AWGN channel	74
Fig. 4.6. Simulated and analytical BER performance of s-PC(17, 12) ² with <i>ordered</i> and conventional shortening in Rayleigh fading channel.....	75
Fig. 4.7. Simulated and analytical BER performance of s-PC(20, 10) ² with <i>ordered</i> and conventional shortening in AWGN channel	76
Fig. 5.1. Asymptotic BER bound and simulated BER values for PC(7, 4) ² in Pulsed Jamming AWGN Channel with $\rho = 0.3$ and $E_b/W_o = 7\text{dB}$	91
Fig. 5.2. Asymptotic BER bound and simulated BER values for PC(7, 4) ² in Pulsed Jamming AWGN Channel with $\rho = 0.05$ and $E_b/W_o = 7\text{dB}$	93
Fig. 5.3. Asymptotic BER bound and simulated BER values for PC(7, 4) ² in Pulsed Jamming Rayleigh Fading Channel with $\rho = 0.3$ and $E_b/W_o = 7\text{dB}$	94
Fig. 5.4. Asymptotic BER bound and simulated BER values for PC(7, 4) ² in Pulsed Jamming Rayleigh Fading Channel with $\rho = 0.05$ and $E_b/W_o = 7\text{dB}$	95
Fig. 5.5. Asymptotic BER bound and simulated BER values for PC(31, 26) ² in Pulsed Jamming AWGN Channel with $\rho = 0.3$ and $E_b/W_o = 7\text{dB}$	97
Fig. 5.6. Asymptotic BER bound and its simulated values for PC(31, 26) ² in Pulsed Jamming Rayleigh Fading Channel with $\rho = 0.5$, $E_b/W_o = 7\text{dB}$	98

List of Tables

Table 2.1. Code Distance and weight distributions of Hamming(7, 4)	16
Table 2.2. Code Distance and weight distributions of e-Hamming(8, 4)	16
Table 3.1. Comparison of True and Approximated $A_{d(s)}$ for Golay code.....	40
Table 3.2. Comparison of True and Approximated $A_{d(s)}$ for Hamming code	41
Table 3.3. Comparison of True and Approximated $A_{d(s)}$ for BCH code	42
Table 3.4. Comparison of True and Approximated $A_{d(p)}$ for p-Hamming(6,4)	46
Table 3.5. Comparison of True and Approximated $A_{d(p)}$ for Hamming(7,4,3)	48
Table 3.6. Comparison of True and Approximated $A_{d(p)}$ for Hamming(15,11,3)	48
Table 3.7. Comparison of True and Approximated $A_{d(p)}$ BCH(15,7,5)	49
Table 3.8. Comparison of True and Approximated $A_{d(p)}$ for BCH(15,5,7)	50
Table 4.1. Comparison of d_{min} of conventional shortened codes and theoretical upper-bound values of d_{min}	61
Table 4.2. Information symbol positions to shorten to achieve an increased d_{min} for Hamming and BCH codes with $n = 15$ and 31	69

Conventions of Symbols Notations

- Small capital letter and italic, e.g. x , to represent a scalar symbol or variable.
 - Bold small capital letter and italic e.g. $\mathbf{x}$, to represent a vector.
 - Bold big capital letter enclosed with square brackets, e.g. $[\mathbf{X}]$, to represent a matrix.
 - Big capital letter enclosed with curly brackets and italic, e.g. $\{\mathbf{x}\}$, to represent a set.
-

Chapter 1

Introduction

1.1 Literature Review and Motivation

Product codes have become an important class of error correction codes in recent years. They have been shown to exhibit good bit-error-rate (BER) performance with reasonable decoding complexity. In particular, using a modified-Chase algorithm, Pyndiah [1][2][3] has shown that more than 98% of channel capacity can be achieved with high-code-rate product codes using only 4 iterations on the Gaussian channel and that the BER slope for the Rayleigh channel without using channel state information is as steep as the Gaussian channel. Moreover, unlike convolutional-based Turbo codes, product codes suffer from lower error floors. They have been adopted in the IEEE 802.16 standard for wireless metropolitan area networks (WMAN).

The modified chase algorithm by Pyndiah [1], however has several limitations. Firstly, it uses two heuristic parameters, α and β which require optimization through repeated simulations. Secondly, the extrinsic information in the algorithm has been normalized to a mean absolute value of one. Hence, it does not allow for varying channel conditions. However, with both sequences α and β made adaptive by [4], the algorithm is applicable to various codes, modulation schemes and channel conditions. Also, [5] has

developed an alternative approach to estimating β using the distance properties of the component codes.

Another limitation of modified Chase algorithm by [1] is that the decoding complexity becomes very large when the minimum Hamming distance of the component codes for the product code is large. Hence, with the aim of reducing the complexity, research [6][7] has been done on list-based soft-in-soft-out (SISO) decoding algorithms, which calculate extrinsic information using a list of codewords. The “box and match” decoding algorithm by [8] is also able to reduce the computational complexity of the ordered statistic decoding algorithm at the expense of increased memory requirements.

In digital communication systems with FEC (forward error correction) involving product codes, code extension and shortening, or a combination of both, are commonly used [9]. A code can be shortened by setting certain information bits to zero during encoding and not transmitting these nullified information bits [10]. An extended code can be constructed by appending a single parity check bit on all the bits in the code word [10]. Shortening of a code is usually used to flexibly adjust the code length and code rate, while code extension is used to improve the BER performance with only a slight decrease in the code rate. Similar to code shortening, puncturing is also usually used to adjust the code length and code rate. However, unlike code shortening, code puncturing does not nullify the information symbols and instead stops the transmission of some parity symbols, causing an increase in the code rate.

The BER performance of a FEC code relies critically on the weight enumerator spectrum of the code, hence much research has been conducted on finding the weight enumerators of good codes such as Hamming [12], BCH [12][13][13] and product

codes[11]. Although the full weight spectrum of a Hamming code or BCH code can be found, the full weight spectrum of product code is not known. [14] attempts to give a better estimate of the full weight spectrum of product code. Specifically, it first computes the known (low) weight codewords based on known parameters of the component codes making up the product code; then it estimates the distribution of unknown weight codewords using a binomial approximation. The drawback of the resultant estimated weight spectrum, however, is that the predicted BER at low E_b/N_o tends to be over-estimated.

The weight enumerator research mentioned above pertains to un-extended and un-shortened codes. The effect of code extension on the code weight enumerator can be easily deduced [15], but the same cannot be said of code shortening. In [17], the undetected error probability of shortened Hamming codes is developed, and the weight distributions of shortened Hamming code are found through its dual codes. However, the result in [17] is limited to the class of shortened Hamming code and not other shortened codes such as BCH codes and product codes. As product codes in shortened forms have been considered in standards IEEE802.16a, it is important to know analytically how their BER performance will be affected by code shortening. To our knowledge, no such analytical performance studies have been reported in the literature. Hence, we would like to develop useful weight enumerator expression of shortened code obtained from generic block and product codes.

The conventional shortening scheme performs shortening of a code by nullifying a block of contiguous information symbol positions during encoding and not transmitting these nullified information symbols [10]. We have observed that such shortening

approach is not optimal in terms of the resultant code distance and correspondingly the decoding bit error probability performance. Hence we would like to find new ways of code shortening that can provide better code properties and BER performance.

Regarding the practical decoding algorithm for product code, work has been reported in [17][18][19] on using the Pyndiah's decoding algorithm in Gaussian and Rayleigh channels. In jamming channels, which are of interest to military communications [21], turbo codes and serially concatenated convolutional codes have been investigated [20][21], not product codes. We are hence interested to study the performance of product codes in pulsed jamming channels, in terms of analytical BER bounds as well as the optimality of practical decoding algorithm based on Pyndiah's algorithm. Pulsed jamming channel is a class of jamming channel, which is fairly "powerful" and hence is of interest [21][34].

1.2 Objectives

Following the motivations mentioned earlier, the 3 main objectives of this thesis are:

- a) To derive useful weight enumerator expression for shortened codes and thereafter apply it to obtain the BER bounds for shortened codes.
 - b) To develop a new method of code shortening such that the resultant code properties (minimum Hamming distance and minimum weight enumerator) can be improved in order to produce better BER performance.
 - c) To derive the BER bounds of product codes in pulsed jamming channels, and to extend and optimize Pyndiah's algorithm for such channels.
-

1.3 Major Contributions of the Thesis

The major contributions of the thesis include:

- a) Closed-form weight enumerator expression has been for conventional shortened and shortened-extended block/product codes. It is expressed in known parameters of its original (un-shortened) codes and hence can be easily obtained. It has been verified to be tight for Hamming, BCH and Golay codes. For small codes, the tightness of the proposed weight enumerator is verified with the true weight enumerator values, while for larger codes, the tightness is verified by deriving the associated BER bounds and comparing with Monte Carlo simulation results. The asymptotic coding loss expressions due to code shortening have also been derived.
- b) An improved method of code shortening called *Ordered Shortening* has been developed for systematic codes. Under this new scheme, we propose to shorten a code by nullifying carefully selected information symbol positions which correspond to codewords with the smallest Hamming distances. As such, the shortened codes from this proposed scheme produce larger d_{min} (minimum Hamming distance) and/or smaller A_d (code weight enumerator) values than a *conventional*-shortened code with the same code length and code rate. These improved d_{min} and $A_{d_{min}}$ properties lead to improved BER performance as verified by computer simulations. Furthermore, the d_{min} values of the proposed *ordered shortening* scheme are shown to meet the theoretical upper bound values for some shortened BCH codes.

- c) We have derived the BER bound for pulsed jamming channel with Rayleigh fading. We have also extended and optimized Pyndiah's decoding algorithm to pulsed jamming channels with AWGN and Rayleigh fading. We show that with the proper use of jamming state information and channel state information to obtain the test positions and to compute the extrinsic information, decoding product codes using Pyndiah's algorithm is able to approach the asymptotic BER bound well. We also show that the α weighting factors in Pyndiah's decoding algorithm for large product codes need to be re-optimized in order to produce the best BER results.

1.4 Organization of the Thesis

The thesis is organized in the following way:

Chapter 2 gives an overview of some background theory of linear block codes, product codes, the BER union bounds and the Pyndiah's soft-in-soft-out (SISO) decoding algorithm for product codes.

Chapter 3 derives the weight enumerator of shortened codes. It then makes the use of the weight enumerator to obtain BER bounds of shortened and shortened-extended codes of linear block codes and product codes. The new bounds developed are verified using simulated results. Coding loss expressions are derived.

Chapter 4 introduces a novel code shortening scheme, called "*Ordered Shortening*", and elaborates on its various strength over conventional shortening scheme.

Chapter 5 extends and optimizes the Pyndiah's decoding algorithm to pulsed jamming channels. The BER bounds for such pulsed jamming channels are also derived. Simulations and analytical BER results are compared and discussed.

Chapter 6 concludes the thesis and provides suggestions for future work.

Chapter 2

Background

2.1 Linear Block Codes

2.1.1 Linear Block Codes

A block code with length n and 2^k codewords is called a *linear* (n, k) code if and only if its 2^k code words form a k -dimensional subspace of the vector space of all the n -tuples over the field GF(2) [12]. The encoder of a block code breaks the information sequence into message blocks of k information symbols each. After encoding, each message block is transformed independently into a codeword of n symbols. The set of 2^k codewords of length n is called a (n, k) code. A (n, k) linear block code has code length n , message length k and satisfies the following conditions [23]:

- 1) component-wise modulo-2 sum of two codewords is another codeword and
- 2) the code contains the all-zero codeword.

The code rate r_c is given by k/n .

2.1.2 Matrix Description of Linear Block Codes

Note that the addition and multiplication operations involved in Sections 2.1.2 and 2.1.3 refer to the modulo-2 addition and multiplication.

A (n, k) linear block code can be specified by a set of k linearly independent binary n -tuples $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$. The codewords are obtained as linear combinations of these k n -tuples. The $\mathbf{g}_1, \mathbf{g}_2, \dots, \mathbf{g}_k$ vectors can be arranged as rows of a $k \times n$ matrix $[\mathbf{G}]$ called the generator matrix of the code as follows [23]:

$$[\mathbf{G}] = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1,n} \\ g_{21} & g_{22} & \cdots & g_{2,n} \\ \vdots & \vdots & \vdots & \vdots \\ g_{k,1} & g_{k,2} & \cdots & g_{k,n} \end{bmatrix}.$$

The codeword $\mathbf{e} = [e_1, e_2, \dots, e_n]$ for an information message $\mathbf{m}$ can then be written as

$$\mathbf{e} = \mathbf{m} \cdot [\mathbf{G}] = m_1 \cdot \mathbf{g}_1 + m_2 \cdot \mathbf{g}_2 + \cdots + m_k \cdot \mathbf{g}_k.$$

Fig. 2.1. Structure of Systematic Codeword

A systematic code has the additional structure that the message sequence is itself part of the codeword. The k symbols of the n -tuple codeword are identical to the information sequence and the remaining $(n-k)$ symbols constitute the parity symbols, as shown in Fig. 2.1. The generator matrix for a systematic block code has the following form $[\mathbf{G}] = [\mathbf{P} \quad \mathbf{I}_k]$ where $[\mathbf{P}]$ is a $k \times (n-k)$ matrix and $[\mathbf{I}_k]$ is the $k \times k$ identity matrix. The parity check matrix $[\mathbf{H}]$ of the systematic code is then given by $[\mathbf{H}] = [\mathbf{I}_{n-k} \quad \mathbf{P}^T]$ where $[\mathbf{P}^T]$ is an $(n-k) \times k$ matrix given by the transpose of $[\mathbf{P}]$.

2.1.3 Minimum Hamming distance

The Hamming weight of a codeword is equal to the number of non-zero symbols in the codeword. For linear block codes, the minimum Hamming distance d of the code is given by the smallest Hamming weight of the non-zero codewords. Given the parity check matrix $[\mathbf{H}] = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_l, \dots, \mathbf{h}_n]$ where $\mathbf{h}_l$ represents the l th column of $[\mathbf{H}]$, a linear block code is defined by the set of codewords $\{\mathbf{e} \mid \mathbf{e} \cdot [\mathbf{H}]^T = \mathbf{0}\}$ where $\mathbf{0}$ refers to a zero vector [23]. That is, a codeword $\mathbf{e}$ satisfies $e_1 \mathbf{h}_1 + e_2 \mathbf{h}_2 + \dots + e_l \mathbf{h}_l + \dots + e_n \mathbf{h}_n = \mathbf{0}$ where e_l is the l th symbol of the codeword $\mathbf{e}$. As such, the non-zero symbol positions of the codeword $\mathbf{e}$ correspond to the columns of $[\mathbf{H}]$ that sum to $\mathbf{0}$. Hence, the minimum Hamming distance of a linear block code is also given by the smallest number of columns of the matrix $[\mathbf{H}]$ that sum to $\mathbf{0}$ [23]. This minimum Hamming distance parameter of a block code is important as it determines the error correcting and error detecting capability of a code.

Note that a linear block code can also be represented by the following convention: $(n, k, d_{\min})$ where $d_{\min}$ is the minimum Hamming distance of the code.

2.1.4 Weight Enumerator

For a block code having a Hamming distance distribution $\{d_{\min}, \dots, d_i, \dots, d_{\max}\}$, where the set $\{d_{\min}, \dots, d_i, \dots, d_{\max}\}$ denotes all possible Hamming distances that the code has, it has a correspondingly weight spectrum $\{A_{d_{\min}}, \dots, A_{d_i}, \dots, A_{d_{\max}}\}$ where A_{d_i} is the number of all codewords with Hamming weight d_i . The weight spectrum of a block code is useful as it is used in obtaining its BER bound for a code.

It has been observed that the binomial distribution is a close approximation to the weight distributions of most binary (n, k) linear codes [13]. In [24] and [25], the following binomial approximation is used:

$$A_d = \begin{cases} 1, & d = 0, d = n \\ \lambda \binom{n}{d}, & d_{\min} \leq d \leq n - d_{\min} \\ 0, & \text{elsewhere} \end{cases} \quad (2.1)$$

This distribution is palindromic with $A_d = A_{n-d}$. λ is chosen such that the total number of codewords of a code is satisfied i.e. $\sum_d A_d = 2^k$. Therefore λ is given by

$$\lambda = \frac{2^k - 2}{\sum_{d=d_{\min}}^{n-d_{\min}} \binom{n}{d}}. \quad (2.2)$$

Example

The weight distribution for a Hamming (7, 4) is given by $A_0 = 1$, $A_3 = 7$, $A_4 = 7$ and $A_7 = 1$ with $\lambda = 0.2$.

2.1.5 Families of Linear Block Codes Considered in this Thesis

In this thesis, we shall consider the families of Hamming, BCH and their product codes in the context of code shortening and Turbo decoding in jamming channels. Their essential features are summarized below.

2.1.5.1 Hamming Codes

Hamming codes are characterized by the following parameters:

$$\begin{aligned}
\text{Block length:} & n = 2^m - 1 \\
\text{No of message symbols:} & k = 2^m - 1 - m \\
\text{No of parity symbols:} & n - k = m \\
\text{Minimum Hamming distance:} & d_{\min} = 3
\end{aligned} \tag{2.3}$$

where $m \geq 3$. Hamming codes are single-error correcting binary *perfect*¹ codes.

2.1.5.2 BCH Codes

BCH codes are characterized by the following parameters:

$$\begin{aligned}
\text{Block length:} & n = 2^m - 1 \\
\text{No of message symbols:} & n - k \leq mt \\
\text{Minimum Hamming distance:} & d_{\min} = 2t + 1
\end{aligned} \tag{2.4}$$

where m and t are positive integers, $m \geq 3$ and $t \geq 2^m - 1$. The error correcting capability of the code is also given by t .

2.2 Product Codes

2.2.1 Product Codes

Product codes are serially concatenated codes built up using two or more short block codes. Fig. 2.2 shows the encoder of a product code C^P made up of two systematic linear block codes $C^h(n^h, k^h, d_{\min}^h)$ and $C^v(n^v, k^v, d_{\min}^v)$ called the horizontal and vertical component codes respectively.

¹ A perfect code is a code which satisfies the Hamming bound $2^{n-k} \geq \sum_{l=0}^t \binom{n}{l}$ with equality, where t is the error correcting capability of the code.

Fig. 2.2. Encoder of a Product Code $\mathbf{C}^P$

The product code $\mathbf{C}^P(n^P, k^P, d^P_{\min}) = \mathbf{C}^h \otimes \mathbf{C}^v$ is obtained by [1]:

- 1) placing $(k^v \times k^h)$ information bits in an array of k^v rows and k^h columns;
- 2) encoding the k^v rows using code $\mathbf{C}^h$ row by row;
- 3) encoding the n^h columns using code $\mathbf{C}^v$ column by column.

Fig. 2.3 shows the resultant 2-dimensional product code. It can be seen that all rows of the product code are codewords of $\mathbf{C}^h$ and all columns of the product code are codewords of $\mathbf{C}^v$.

Fig. 2.3. Structure of Product Code $\mathbf{C}^P$

The parameters of product code $\mathbf{C}^p$ are [1]:

- 1) code length: $n^p = n^v \times n^h$,
- 2) message length: $k^p = k^v \times k^h$ and
- 3) minimum Hamming distance: $d_{\min}^p = d_{\min}^v \times d_{\min}^h$.

Hence product code is capable of building very long block codes with large minimum Hamming distance $d_{\min}^p$ using relatively short codes.

2.2.2 Weight Enumerator of Product codes

Unlike block codes, as the dimensions of product codes are higher, the full distance spectrum and weight enumerator are not easily known. However, for the special case where both component codes are binary Hamming codes of length n^h and n^v , the following results have been found [11]:

For $d \leq 17$, the number A_d of weight d codewords in the product of binary Hamming codes of length n^h and length n^v is given by the following formulas:

$$A_d = \begin{cases} 1, & \text{if } d = 0 \\ A_3(n^h) \cdot A_3(n^v), & \text{if } d = 9 \\ A_3(n^h) \cdot A_4(n^v) + A_4(n^h) \cdot A_3(n^v), & \text{if } d = 12 \\ A_3(n^h) \cdot A_5(n^v) + A_5(n^h) \cdot A_3(n^v), & \text{if } d = 15 \\ 19A_4(n^h) \cdot A_4(n^v), & \text{if } d = 16 \\ 36A_4(n^h) \cdot A_4(n^v), & \text{if } d = 17 \\ 0, & \text{otherwise} \end{cases} \quad (2.5)$$

where

$$\begin{aligned} A_3(n) &= n(n-1)/6 \\ A_4(n) &= n(n-1)(n-3)/24 \\ A_5(n) &= n(n-1)(n-3)(n-7)/120 \end{aligned} \quad (2.6)$$

2.3 Code Shortening/Extension

2.3.1 Extended Code

Given a linear binary (n, k) code, an extended $(n+1, k)$ code can be constructed by appending a single parity check bit on all the bits in the code word [10]. Correspondingly, the code rate is reduced to $k/(n+1)$. With code extension, all the odd Hamming distance becomes the next even Hamming distance i.e. $d+1$. Therefore the typically odd minimum Hamming distances of Hamming and BCH codes become even. Also the weight enumerator of the extended code is now given by the summation of A_d with odd d and A_d with the next even d (i.e. $d+1$) i.e. $A_d + A_{d+1}$. Take the extended-Hamming (e-Hamming) $(8, 4)$ as an example, Tables 2.1 and 2.2 shows the Hamming distance spectrum and weight enumerator of Hamming(7, 4) before and after extension respectively.

Table 2.1. Code Distance and weight distributions of Hamming(7, 4)

d	A_d
0	1
3	7
4	7
7	1

Table 2.2. Code Distance and weight distributions of e-Hamming(8, 4)

d	A_d
0	1
4	14
8	1

2.3.2 Shortened Code

A linear (n, k) code consisting of k information bits and $(n-k)$ check bits can be shortened into a $(n-s, k-s)$ linear code by setting the first s information bits to zero during encoding and not transmitting these nullified information bits [10]. Hence the length of the transmitted codeword is *shortened* to $(n-s)$. Correspondingly, the code rate is reduced to $(k-s)/(n-s)$. Also, the minimum Hamming distance of the shortened code is at least as large as the minimum distance of the original un-shortened code [10].

2.4 Decoding BER Union Bounds

2.4.1 Decoding BER Union Bounds in AWGN Channel

Given a (n, k) binary linear code with weight enumerator A_d , code-rate $r_c = k/n$, signal energy per bit E_b and single-sided power spectral density N_o , the soft decoding bit error probability P_b of this code in an *Additive-White-Gaussian-Noise* (AWGN) channel, assuming coherent binary/quadrature phase shift keying (B/QPSK) modulation, is upper-bounded by [14] [23]

$$P_b \leq \sum_d \frac{d}{n} A_d P_d(d, r_c) \quad (2.7)$$

where

$$P_d(d, r_c) = Q\left(\sqrt{2dr_c \frac{E_b}{N_o}}\right). \quad (2.8)$$

$P_d(d, r_c)$ is called the pair-wise error probability [23]. It is the probability that the decoder makes a wrong decision by selecting another codeword, instead of the actual transmitted codeword, as the decoded sequence.

2.4.2 Decoding BER Union Bounds in Rayleigh Fading Channel

When channel state information is available to the channel decoder, the average bit error probability of the code in memoryless Rayleigh fading channel is upper-bounded by

$$\bar{P}_b \leq \sum_d \frac{d}{n} A_d \bar{P}_d(d, r_c) \quad (2.9)$$

where $\bar{P}_d(d, r_c)$ is the average pairwise error probability.

$\bar{P}_d(d, r_c)$ can be found by conditioning on the fading coefficient vector $\mathbf{a} = [a_1, \dots, a_j, \dots, a_n]$ where a_j is the fading amplitude modeled with a Rayleigh probability density function, and then averaging this pairwise error probability over the pdf $p_a(\mathbf{a})$ of $\mathbf{a}$. Assuming that the fading coefficients are independent, the conditioned pairwise error probability is given by [26]

$$\bar{P}_d(d, r_c | \mathbf{a}) = Q\left(\sqrt{2r_c \frac{E_b}{N_o} \sum_{l=1}^d a_l^2}\right). \quad (2.10)$$

Therefore the average pairwise error probability is

$$\bar{P}_d(d, r_c) = \int_{\mathbf{a}} p_a(\mathbf{a}) \bar{P}_d(d, r_c | \mathbf{a}) d\mathbf{a} \quad (2.11)$$

Since $\mathbf{a}$ is a multi-dimensional variable,

$$\bar{P}_d(d, r_c) = \int_{a_1} \dots \int_{a_d} p_a(a_1, a_2, \dots, a_d) \bar{P}_d(d, r_c | (a_1, a_2, \dots, a_d)) da_1, \dots, da_d. \quad (2.12)$$

For memoryless fading channel, each a_l is independent, hence

$$p_a(\mathbf{a}) = p_a(a_1, a_2, \dots, a_d) = \prod_{l=1}^d p(a_l) \quad (2.13)$$

where

$$p(a_l) = 2a_l e^{-a_l^2} \quad (2.14)$$

is the probability density function (pdf) for the Rayleigh distribution.

It is noted that $Q(x)$ can be expressed in an alternative form [26] given by

$$Q(x) = \frac{1}{\pi} \int_0^{\pi/2} e^{-\frac{x^2}{2\sin^2\phi}} d\phi. \quad (2.15)$$

Using (2.13), (2.14) and (2.16) in (2.12), it can be shown that the d -dimensional integral for $\bar{P}_d(d, r_c)$ reduces to a product of integrals over each a_l [26] to give:

$$\bar{P}_d(d, r_c) = \frac{1}{\pi} \int_0^{\pi/2} \left[\frac{\sin^2 \phi}{r_c \frac{E_b}{N_0} + \sin^2 \phi} \right]^d d\phi \quad (2.16)$$

Hence, with the following parameters of a code: weight distribution A_d , n and k , the BER performance bound of a linear block code can be computed via (2.7) or (2.9).

2.5 Pyndiah's SISO Decoding Algorithm of Product Codes for AWGN Channel

Pyndiah [1] has proposed a soft-in-soft-out (SISO) iterative decoding algorithm for product codes and shown that it is able to achieve optimum BER performance for high-rate component codes (with low minimum Hamming distance) such as Hamming codes by sequentially decoding the rows and columns of the product code with feedback. Other list-based decodings have also been developed by [7][Soft-Input Soft-Output List-Based Decoding Algorithm] [8][On Soft-Input Soft-Output Decoding Using “Box and Match” Techniques] and shown to be capable of achieving good performance and complexity tradeoff when lower rate component codes (with high minimum Hamming distance) are used.

In this section, we shall review the essential steps and underlying principles of Pyndiah's algorithm in the next sections. Because each row or column of the product code is a linear block code (see Section 2.2), we shall consider the soft decoding algorithm of linear block code first before introducing the soft decoding algorithm of the product code.

2.5.1 Soft Decoding of Linear Block Code

In the soft decoding of the linear block code, we review the following decoding schemes [1]:

- Maximum-Likelihood decoding(MLD)
- Chase Decoding

2.5.1.1 AWGN Channel Model

Fig. 2.4 shows the model for an AWGN channel. Consider the transmission of binary phase shift keying (BPSK) modulated vector $\mathbf{x} = (x_1, \dots, x_j, \dots, x_n)$ of the codeword $\mathbf{e} = (e_1, \dots, e_j, \dots, e_n)$ encoded by a linear block code $\mathbf{C}(n, k, d_{\min})$ on an Additive White Gaussian Noise (AWGN) channel. The mappings of symbols $0 \rightarrow -1$ and $1 \rightarrow +1$ are assumed for $\mathbf{e} \rightarrow \mathbf{x}$.

Fig. 2.4. AWGN Channel Model

At the receiver, the received signal $\mathbf{r} = (r_1, \dots, r_j, \dots, r_n)$ is given by

$$\mathbf{r} = \mathbf{x} + \mathbf{n} \quad (2.17)$$

where $\mathbf{n} = (n_1, \dots, n_j, \dots, n_n)$ is a noise vector with each noise sample having mean zero and variance σ_n^2 . The pdf of each AWGN noise sample noise n_j is

$$p(n_j) = \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp \left[-\frac{1}{2} \left(\frac{n_j}{\sigma_n} \right)^2 \right]. \quad (2.18)$$

2.5.1.2 Maximum Likelihood Decoding

Using the log-likelihood ratio (LLR) concept, the reliability vector generated from

$\mathbf{r} = (r_1, \dots, r_j, \dots, r_n)$ is given by

$$\Lambda(r_j) = \ln \left(\frac{\Pr \{x_j = +1 | r_j\}}{\Pr \{x_j = -1 | r_j\}} \right) = \frac{2}{\sigma_n^2} r_j. \quad (2.19)$$

Note that if a stationary channel is considered, the reliability of r_j can be simplified to

$|r_j|$ as the factor $2/\sigma_n^2$ is constant for all received symbols.

The hard detector's output, $\mathbf{y}$, of $\mathbf{r}$ is then given by:

$$y_j = \begin{cases} 0 & \text{if } \Lambda(r_j) < 0 \\ 1 & \text{if } \Lambda(r_j) \geq 0 \end{cases} \quad (2.20)$$

Using MLD, the optimum decision $\mathbf{d} = (d_1, \dots, d_j, \dots, d_n)$ corresponding to $\mathbf{e}$ is given by [1]

$$\mathbf{d} = \mathbf{c}^i \text{ if } |\mathbf{r} - \mathbf{c}^i|^2 \leq |\mathbf{r} - \mathbf{c}^l|^2 \quad \forall l \in [1, 2^k], \quad l \neq i \quad (2.21)$$

where $\mathbf{c}^i = (c_1^i, \dots, c_j^i, \dots, c_n^i)$ is the i th codeword of $\mathbf{C}$ and

$$|\mathbf{r} - \mathbf{c}^i|^2 = \sum_{j=1}^n (r_j - c_j^i)^2 \quad (2.22)$$

is the squared Euclidean distance between $\mathbf{r}$ and $\mathbf{c}^i$. The optimum decision $\mathbf{d}$ is given by the codeword with the minimum squared Euclidean distance between itself and $\mathbf{r}$. As can be seen, the computing complexity of ML decoding increases exponentially with k when an exhaustive search is used for $\mathbf{d}$. Hence, the exhaustive search is not a realistic solution for large k . Therefore, the sub-optimal Chase algorithm for reducing the number of reviewed codewords is used [1].

2.5.1.3 Chase's Selection of Codewords

Chase observed that at high signal-to-noise ratio (SNR), the probability that the ML codeword $\mathbf{d}$ is located in the sphere of radius $(d_{\min} - 1)$ centered on the hard demodulator output $\mathbf{y} = (y_1, \dots, y_j, \dots, y_n)$ given by (2.20) is very high. Hence, the number of reviewed codewords can be limited to those in the sphere of radius $(d_{\min} - 1)$ centered on $\mathbf{y}$. This set of codewords to be reviewed can be selected using the channel information $|\Lambda(r_j)|$.

The procedure to identify the set of Chase's codewords to be reviewed is as follows:

Step 1: Determine $m = \left\lfloor \frac{d_{\min}}{2} \right\rfloor$ least reliable binary elements of $\mathbf{y}$ with the smallest reliability value of $|\Lambda(r_j)|$.

Step 2: Form a set of test patterns $\{\mathbf{t}^p\}$ by permuting the 2^m possible error patterns identified in step 1.

Step 3: Form the set of test sequences $\{\mathbf{z}^p = \mathbf{y} \oplus \mathbf{t}^p\}$.

Step 4: Perform hard decoding on $\{\mathbf{z}^p\}$ to obtain the set of candidate codewords $\{\mathbf{c}^q\}$.

Therefore, it can be seen that the set of codewords using the exhaustive search has been limited to the set of 2^m candidate codewords. Hence the optimum decoding decision $\mathbf{d}$ can be chosen from $\{\mathbf{c}^q\}$ using (2.21).

2.5.2 Soft Decoding Output of Linear Block Codes

2.5.2.1 Soft Output - Reliability of Decision $\mathbf{d}$

After determining the decoding decision $\mathbf{d}$ of a block code (either a row or a column), the reliability of each components of vector $\mathbf{d}$ is computed in order to generate soft decisions at the output of the decoder. The reliability of decision d_j at the output of the decoder is defined using the log-likelihood ratio (LLR)

$$\Lambda(d_j) = \ln \left(\frac{\Pr\{x_j = +1 | \mathbf{r}\}}{\Pr\{x_j = -1 | \mathbf{r}\}} \right). \quad (2.23)$$

Because $\mathbf{d}$ is one of the 2^k codewords of $\mathbf{C}$, the numerator of (2.23) can be written as following by considering different codewords of $\mathbf{C}$

$$\Pr\{x_j = +1 | \mathbf{r}\} = \sum_{\mathbf{c}^i \in \{\mathbf{s}_j^{+1}\}} \Pr\{\mathbf{x} = \mathbf{c}^i | \mathbf{r}\} \quad (2.24)$$

where $\{\mathbf{s}_j^{+1}\}$ is the set of codewords $\{\mathbf{c}^i\}$ such that $c_j^i = +1$ while the denominator of (2.23) can similarly be written in the form

$$\Pr\{x_j = -1 | \mathbf{r}\} = \sum_{\mathbf{c}^i \in \{\mathbf{s}_j^{-1}\}} \Pr\{\mathbf{x} = \mathbf{c}^i | \mathbf{r}\} \quad (2.25)$$

where $\{\mathbf{s}_j^{-1}\}$ is the set of codewords $\{\mathbf{c}^i\}$ such that $c_j^i = -1$.

By applying Bayes' rule to (2.24) and (2.25), and by assuming that the different codewords are uniformly distributed, equation (2.23) can be written as

$$\Lambda(d_j) = \ln \left(\frac{\sum_{\mathbf{c}^i \in \{\mathbf{s}_j^{+1}\}} p(\mathbf{r} | \mathbf{x} = \mathbf{c}^i)}{\sum_{\mathbf{c}^i \in \{\mathbf{s}_j^{-1}\}} p(\mathbf{r} | \mathbf{x} = \mathbf{c}^i)} \right) \quad (2.26)$$

where

$$p(\mathbf{r} | \mathbf{x} = \mathbf{c}^i) = \left(\frac{1}{\sqrt{2\pi}\sigma_n} \right)^n \exp \left(-\frac{|\mathbf{r} - \mathbf{c}^i|^2}{2\sigma_n^2} \right) \quad (2.27)$$

Combining equation (2.26) and (2.27), the reliability of decision d_j will be

$$\Lambda(d_j) = \ln \left(\frac{\sum_{\mathbf{c}^i \in \{s_j^{+1}\}} \exp \left(-\frac{\|\mathbf{r} - \mathbf{c}^i\|^2}{2\sigma_n^2} \right)}{\sum_{\mathbf{c}^i \in \{s_j^{-1}\}} \exp \left(-\frac{\|\mathbf{r} - \mathbf{c}^i\|^2}{2\sigma_n^2} \right)} \right) \quad (2.28)$$

To simplify the computation, we only consider the two codewords $\mathbf{c}^{+l(j)}$ and $\mathbf{c}^{-l(j)}$, in $\{s_j^{+1}\}$ and $\{s_j^{-1}\}$ respectively, which have minimum Euclidean distance from $\mathbf{r}$.

Hence, (2.28) becomes

$$\Lambda(d_j) = \frac{1}{2\sigma_n^2} \left(\|\mathbf{r} - \mathbf{c}^{-l(j)}\|^2 - \|\mathbf{r} - \mathbf{c}^{+l(j)}\|^2 \right) + \ln \left(\frac{\sum_i A_i}{\sum_i B_i} \right) \quad (2.29)$$

where

$$A_i = \exp \left(\frac{\|\mathbf{r} - \mathbf{c}^{+l(j)}\|^2 - \|\mathbf{r} - \mathbf{c}^i\|^2}{2\sigma_n^2} \right) \text{ with } \mathbf{c}^i \in \{s^{+l(j)}\} \quad (2.30)$$

and

$$B_i = \exp \left(\frac{\|\mathbf{r} - \mathbf{c}^{-l(j)}\|^2 - \|\mathbf{r} - \mathbf{c}^i\|^2}{2\sigma_n^2} \right) \text{ with } \mathbf{c}^i \in \{s^{-l(j)}\}. \quad (2.31)$$

For high SNR, as $\sigma_n \rightarrow 0$, $\sum_i A_i \approx \sum_i B_i$, hence an approximation of $\Lambda(d_j)$ can be

obtained as

$$\Lambda(d_j) \approx \Lambda'(d_j) = \frac{1}{2\sigma_n^2} (\|\mathbf{r} - \mathbf{c}^{-l(j)}\|^2 - \|\mathbf{r} - \mathbf{c}^{+l(j)}\|^2) \quad (2.32)$$

Expanding (2.32) using (2.22), we obtain

$$\Lambda'(d_j) = \frac{2}{\sigma_n^2} \left(r_j + \sum_{l=1, l \neq j}^n r_l c_l^{+1(j)} p_l \right) \quad (2.33)$$

where

$$p_l = \begin{cases} 0, & \text{if } c_l^{+1(j)} = c_l^{-1(j)} \\ 1, & \text{if } c_l^{+1(j)} \neq c_l^{-1(j)} \end{cases} \quad (2.34)$$

Normalizing $\Lambda'(d_j)$ with respect to the constant $2/\sigma_n^2$, we obtain

$$r'_j = \frac{\sigma_n^2}{2} \Lambda'_j = r_j + w_j \quad (2.35)$$

where the extrinsic information is given by

$$w_j = \sum_{l=1, l \neq j}^n r_l c_l^{+1(j)} p_l \quad (2.36)$$

2.5.2.2 Extrinsic Information

As shown in (2.32), computing the reliability of decoding decision d_j at the output of the soft-input decoder needs two code words $c^{+1(j)}$ and $c^{-1(j)}$. The soft decision $\mathbf{d}$ from Chase algorithm is one of these two code words. Assuming that another codeword $\mathbf{c}$ can be found from the candidate codewords $\{\mathbf{c}^q\}$ of the Chase algorithm to have minimum Euclidean distance from $\mathbf{r}$ with $c_j \neq d_j$, it can be viewed as a competing code word of $\mathbf{d}$. Given codeword $\mathbf{c}$ and $\mathbf{d}$, by normalizing (2.32) with respect to the constant $2/\sigma_n^2$, the soft output is given by

$$r'_j = \left(\frac{|r-c|^2 - |r-d|^2}{4} \right) d_j \quad (2.37)$$

But in some cases, a competing codeword c cannot be found from $\{c^q\}$. In this case, the size of the search space scanned by the Chase algorithm should be increased and the computing complexity will also increase. To alleviate this problem, Pyndiah proposed a simple solution for computing the soft output:

$$r'_j = \beta d_j \quad (2.38)$$

where $\beta \geq 0$ can be manually obtained a-priori.

Therefore the extrinsic information can be expressed as

$$w_j = r'_j - r_j = \begin{cases} \left(\frac{|r-c|^2 - |r-d|^2}{4} d_j \right) - r_j & \text{if } c \text{ exists} \\ \beta \times d_j - r_j & \text{if } c \text{ does not exist} \end{cases} \quad (2.39)$$

2.5.3 Turbo Decoding of Product Codes

Let us consider the decoding of the rows and columns of a product code transmitted on a Gaussian channel using BPSK signaling. On receiving the code matrix $[\mathbf{R}]$ corresponding to a transmitted codeword $[\mathbf{E}]$, the first decoder performs SISO decoding of the rows (or columns) of $[\mathbf{R}]$ using the Chase algorithm. The soft output is computed using (2.37) and (2.38) while the extrinsic information $[\mathbf{W}]$ is computed using (2.39). The soft input for the decoding of the next columns (or rows) is then given by

$$[\mathbf{R}(m+1)] = [\mathbf{R}] + \alpha(m)[\mathbf{W}(m)] \quad (2.40)$$

where the index m denotes the m^{th} decoding step, and α is a scaling factor which takes into account the fact that the standard deviation of samples in $[\mathbf{R}]$ and $[\mathbf{W}]$ are different. The standard deviation of the extrinsic information is very high in the first decoding step and decreases as the decoding iterations proceed. This scaling factor α is used to reduce the effect of the extrinsic information in the first decoding steps when the BER is relatively high. It takes a small value in the early decoding steps and increases as the BER improves. The component decoding procedure described above is depicted graphically in Fig. 2.5. It is cascaded for successive row/column decoding of the product code to perform iterative decoding.

Fig. 2.5. Block diagram of component decoder in Pyndiah's decoder

The values of weighting factor α used by Pyndiah at the m th decoding step are given by

$$\alpha(m) = [0.0, 0.2, 0.3, 0.5, 0.7, 0.9, 1.0, 1.0]. \quad (2.41)$$

The values of reliability factor β used by Pyndiah at the m th decoding are given by

$$\beta(m) = [0.2, 0.4, 0.6, 0.8, 1.0, 1.0, 1.0, 1.0]. \quad (2.42)$$

These α and β values are used optimized manually for product codes composed of Hamming and BCH codes for AWGN and Rayleigh fading channels [1].

As from (2.41) and (2.42), the pre-computed sequences of α and β allow for a maximum of 4 iterations (which we shall use in our simulations). For greater number of iterations, the adaptive versions of α and β in [4][5][35] may be used.

2.6 Pyndiah's SISO Decoding Algorithm of Product Code for Rayleigh Fading Channel

The decoding process for product codes in a Rayleigh fading channel is similar to that in Section 2.5 except for small changes. Hence in this section, we shall only present the changes, assuming a coherently detected, fully interleaved Rayleigh fading channel.

2.6.1 Rayleigh Flat Fading Channel Model

Fig. 2.6. Rayleigh Flat Fading Channel Model

With the same definitions of x and n as in Section 2.5.1.1, Fig. 2.6 shows the model for a Rayleigh flat fading channel. Given the above channel, the received signal is given by

$$r = x[A] + n \quad (2.43)$$

where $\mathbf{A} = \begin{bmatrix} a_1 & 0 & \cdots & 0 \\ 0 & a_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_n \end{bmatrix}$ is a diagonal matrix with the fading coefficient a_i . Each

$a_i = a_I + ja_Q$ where a_I and a_Q are Gaussian distributed with their variance normalized to

$\frac{1}{2}$ such that the magnitude $a_i = \sqrt{a_I^2 + a_Q^2}$ is Rayleigh distributed with $E[|a_i|^2] = 1$.

2.6.2 Decoding with Channel State Information

Assuming perfectly known $\mathbf{A}$ at the receiver, as opposed to (2.19), the log-likelihood ratio (LLR) of decision r_j of the Rayleigh fading channel is given by

$$\begin{aligned} \Lambda(r_j) &= \ln \left(\frac{\Pr\{x_j = +1, a_j | r_j\}}{\Pr\{x_j = -1, a_j | r_j\}} \right) \\ &= \ln \left(\frac{\Pr\{r_j | x_j = +1, a_j\}}{\Pr\{r_j | x_j = -1, a_j\}} \right) \\ &= \frac{2}{\sigma_n^2} a_j r_j \end{aligned} \quad (2.44)$$

where

$$\Pr\{r_j | x_j, a_j\} = \frac{1}{2\pi\sigma_n^2} \exp\left(-\frac{|r_j - a_j x_j|^2}{2\sigma_n^2}\right). \quad (2.45)$$

It can be shown that reliability of decision d_j at the output of the decoder, as defined using the log-likelihood ratio (LLR), can be approximated by

$$\Lambda(d_j) \approx \Lambda'(d_j) = \frac{1}{2\sigma_n^2} (|\mathbf{r} - \mathbf{c}^{-1(j)}[\mathbf{A}]|^2 - |\mathbf{r} - \mathbf{c}^{+1(j)}[\mathbf{A}]|^2) \quad (2.46)$$

Expanding (2.46) using $\left| \mathbf{r} - \mathbf{c}^i[\mathbf{A}] \right|^2 = \sum_{l=1}^n \left(r_l - c_l^i a_l \right)^2$, we obtain

$$\Lambda'(d_j) = \frac{2}{\sigma_n^2} \left(a_j r_j + \sum_{l=1, l \neq j}^n a_l r_l c_l^{+1(j)} p_l \right) \quad (2.47)$$

where

$$p_l = \begin{cases} 0, & \text{if } c_l^{+1(j)} = c_l^{-1(j)} \\ 1, & \text{if } c_l^{+1(j)} \neq c_l^{-1(j)} \end{cases} \quad (2.48)$$

Normalizing $\Lambda'(d_j)$ with respect to the constant $2/\sigma_n^2$, we obtain

$$r'_j = a_j r_j + w_j \quad (2.49)$$

where the extrinsic information is given by

$$w_j = \sum_{l=1, l \neq j}^n a_l r_l c_l^{+1(j)} p_l \quad (2.50)$$

Also, in contrast with (2.37), the soft output of the decoder in a Rayleigh fading channel is given by

$$r'_j = \left(\frac{\left| \mathbf{r} - \mathbf{c}[\mathbf{A}] \right|^2 - \left| \mathbf{r} - \mathbf{d}[\mathbf{A}] \right|^2}{4} \right) d_j. \quad (2.51)$$

where both $\mathbf{c}[\mathbf{A}]$ and $\mathbf{d}[\mathbf{A}]$ have the minimum Euclidean distance from $\mathbf{r}$ from the set of candidates codewords $\{\mathbf{c}^q\}$ and that $c_j \neq d_j$.

It is interesting to note that if the fading coefficient matrix $\mathbf{A}$ is multiplied on $\mathbf{r}$ instead of on $\mathbf{c}$ and $\mathbf{d}$, the same soft output as (2.47) will result:

$$\begin{aligned}\Lambda(d_j) &\approx \Lambda'(d_j) = \frac{1}{2\sigma_n^2} (|r[\mathbf{A}] - \mathbf{c}^{-1(j)}|^2 - |r[\mathbf{A}] - \mathbf{c}^{+1(j)}|^2) \\ &= \frac{2}{\sigma_n^2} \left(a_j r_j + \sum_{l=1, l \neq j}^n a_l r_l c_l^{+1(j)} p_l \right)\end{aligned}$$

Therefore, the same soft output of the decoder can also be found by

$$r_j = \left(\frac{|r[\mathbf{A}] - \mathbf{c}|^2 - |r[\mathbf{A}] - \mathbf{d}|^2}{4} \right) d_j \quad (2.50)$$

2.6.3 Decoding Without Channel State Information

When $[\mathbf{A}]$ is not known at the receiver, the log-likelihood ratio (LLR) of decision r_j of the Rayleigh fading channel is given by (2.19) and the soft output of the decoder is given by (2.37).

Chapter 3

Effect of Code Shortening on Decoding BER Performance

As shown earlier in Section 2.4, the weight distribution of a code enables the prediction of BER through the union bounds. However, when a code is shortened, its weight distribution will change and its BER performance will be affected. Hence, in this section, we shall derive the approximated weight enumerator for a shortened code and thereafter apply this approximated weight enumerator to the BER bounds.

Note that in this chapter we refer to the conventional way of code shortening as described in Section 2.3.2.

3.1 Weight Enumerator of Shortened Code

3.1.1 Approximation to the Weight Distribution of Shortened Binary Linear Codes

As mentioned in Section 2.1.4, it has been observed that the binomial distribution is a close approximation to the weight distributions of most binary (n, k) linear codes. For a

shortened code with s number of nullified information symbols, we also assume that its weight distribution is binomial-like (to be verified later).

Following (2.1) and denoting the maximum Hamming distance of a shortened code as $d_{\max}$, the weight enumerator of a shortened code $A_{d(s)}$ is then given by

$$A_{d(s)} = \begin{cases} 1, & d = 0 \\ \delta \binom{n-s}{d}, & d_{\min} \leq d \leq d_{\max} \\ 0, & \text{elsewhere} \end{cases} \quad (3.1)$$

where

$$\delta = \frac{2^{k-s} - 1}{\sum_{d=d_{\min}}^{d_{\max}} \binom{n-s}{d}}. \quad (3.2)$$

In (3.2), δ is chosen such that $\sum_d A_{d(s)} = 2^{k-s}$ because the binomial approximation is applied to all its 2^{k-s} shortened codewords except for the all-zero codeword.

Example

To illustrate with a simple example, the weight distribution from (3.1) for a shortened-Hamming (6, 3) is given by $A_0 = 1$, $A_3 = 4$ and $A_4 = 3$ with $\delta = 0.2$. As shown in Fig. 3.1, this estimated weight distribution agrees exactly with its true weight distribution. Note that as the shortened code does not contain the all-one codeword, the weight distribution is no longer palindromic (i.e. $A_d \neq A_{n-d}$)

Fig. 3.1. Comparison of Approximated A_d and true A_d of s-Hamming(6, 3) with $s = 1$

Fig. 3.2 and 3.3 show the comparison of A_d obtained from the binomial approximation (3.1) with its true A_d for shortened-BCH(28, 13) with $s = 3$ and shortened-Hamming(25, 20) with $s = 6$ respectively. Both figures show that (3.1) gives a reasonable approximation of the weight distribution of the shortened codes. They also validate our original assumption that shortened codes have binomial-like weight distribution as (3.1).

Fig. 3.2. Comparison of Approximated A_d and true A_d of s-BCH(28,13) with $s=3$

Fig. 3.3. Comparison of Approximated A_d and true A_d of s-Hamming(25,20) with $s=6$

It would be useful to express $A_{d(s)}$ in term of A_d as this will allow us to compute easily the weight distribution of the shortened codes based on the known weight distributions of the un-shortened codes.

We define a scaling factor

$$f_{s,d} = \frac{A_{d(s)}}{A_d}. \quad (3.3)$$

Using (2.1) and (3.1), $f_{s,d}$ can then be expressed as

$$f_{s,d} = \begin{cases} 1 & , \text{ if } d=0 \\ \frac{(2^{k-s}-1)}{(2^k-2)} \cdot \frac{\binom{n-s}{d}}{\binom{n}{d}} \cdot \frac{\sum_{d=d_{\min}}^{d_{\max}} \binom{n}{d}}{\sum_{d=d_{\min}}^{d_{\max}} \binom{n-s}{d}} & , \text{ if } (n-s) \geq d \\ 0 & , \text{ if } (n-s) < d \end{cases} \quad (3.4)$$

where n and d are parameters of the original (un-shortened) code and s is the number of nullified information symbols.

Define the set of code distances which excludes the $d = 0$ and $d = n$ terms of an un-shortened code to be $\{d_{ori}\}$. Then

$$\sum_{d=d_{\min}}^{d_{\max}} \binom{n}{d} \equiv \sum_{d \in \{d_{ori}\}} \binom{n}{d}. \quad (3.5)$$

Making use of the known result [33]

$$\sum_{d=0}^n \binom{n}{d} = 2^n, \quad (3.6)$$

we have

$$\sum_{d \in \{d_{ori}\}} \binom{n}{d} = 2^n - \sum_{d \notin \{d_{ori}\}} \binom{n}{d} \quad (3.7)$$

Similarly, we define the set of code distances which excludes the $d = 0$ term of a shortened code to be $\{d_{sh}\}$. Then

$$\sum_{d=d_{min}}^{d_{max}} \binom{n-s}{d} \equiv \sum_{d \in \{d_{sh}\}} \binom{n-s}{d}. \quad (3.8)$$

Similarly, making use of (3.6), we obtain

$$\sum_{d \in \{d_{sh}\}} \binom{n-s}{d} = 2^{n-s} - \sum_{d \notin \{d_{sh}\}} \binom{n-s}{d} \quad (3.9)$$

Therefore,

$$\begin{aligned} \frac{\sum_{d=d_{min}}^{n-d_{min}} \binom{n}{d}}{\sum_{d=d_{min}}^{n-d_{min}} \binom{n-s}{d}} &\equiv \frac{2^n - \sum_{d \notin \{d_{ori}\}} \binom{n}{d}}{2^{n-s} - \sum_{d \notin \{d_{sh}\}} \binom{n-s}{d}} \\ &= \frac{2^n \left(1 - \frac{\sum_{d \notin \{d_{ori}\}} \binom{n}{d}}{2^n} \right)}{2^{n-s} \left(1 - \frac{\sum_{d \notin \{d_{sh}\}} \binom{n-s}{d}}{2^{n-s}} \right)} \end{aligned} \quad (3.10)$$

When n is large, $\frac{\sum_{d \notin \{d_{ori}\}} \binom{n}{d}}{2^n} \rightarrow 0$ and $\frac{\sum_{d \notin \{d_{sh}\}} \binom{n-s}{d}}{2^{n-s}} \rightarrow 0$.

Therefore, (3.10) is approximately equal to

$$\frac{\sum_{d=d_{\min}}^{n-d_{\min}} \binom{n}{d}}{\sum_{d=d_{\min}}^{n-d_{\min}} \binom{n-s}{d}} \approx \frac{2^n}{2^{n-s}} = 2^s \quad (3.11)$$

With (3.11), and since n and k are usually large, (3.4) can be simplified to

$$f_{s,d} = \begin{cases} 1 & , \text{ if } d = 0 \\ \frac{\binom{n-s}{d}}{\binom{n}{d}} & , \text{ if } (n-s) \geq d \\ 0 & , \text{ if } (n-s) < d \end{cases} \quad (3.12)$$

Therefore,

$$A_{d(s)} \approx \text{Round}(f_{s,d} A_d) \quad (3.13)$$

where Round(.) refers to rounding of the value(.) to the nearest integer.

3.1.2 Comparison of True and Approximated weight enumerator of Shortened Code

Tables 3.1, 3.2 and 3.3 compare the true and approximated $A_{d(s)}$ values for shortened Golay, Hamming and BCH codes with different amount of shortening s . Disagreements between the true and approximated values are highlighted in bold italic. We can see that there is no difference in the approximated and true $A_{d(s)}$ for shortened Hamming and BCH codes with $s = 1$ and 2 and, for shortened Golay codes $s = 1$ to 7 and $s = 9$. These results show that (3.13) gives tight estimates to the true $A_{d(s)}$. More comparisons for larger Hamming and BCH codes are shown in Appendices A and B. Similarly, they show that

(3.13) gives good estimate to the true $A_{d(s)}$.

Table 3.1. Comparison of True and Approximated $A_{d(s)}$ for Golay code

Golay(23, 12)			d							
			0	7	8	11	12	15	16	23
s	1-7	True $A_{d(s)}$	Agree Exactly							
		Approx. $A_{d(s)}$								
	8	True $A_{d(s)}$	1	7	6	1	1	0	0	0
		Approx. $A_{d(s)}$	1	7	7	1	0	0	0	0
	9	True $A_{d(s)}$	Agree Exactly							
		Approx. $A_{d(s)}$								

Table 3.2. Comparison of True and Approximated $A_{d(s)}$ for Hamming code

Hamming(7, 4)			d											
			0	3	4	7								
s	1-2	True $A_{d(s)}$ Approx. $A_{d(s)}$	Agree Exactly											
Hamming(15, 11)			d											
			0	3	4	5	6	7	8	9	10	11	12	15
s	1-2	True $A_{d(s)}$	Agree Exactly											
		Approx. $A_{d(s)}$												
	3	True $A_{d(s)}$	1	17	38	44	52	54	33	12	4	1	0	0
		Approx. $A_{d(s)}$	1	17	38	44	52	53	33	12	4	1	0	0
	4	True $A_{d(s)}$	1	13	25	25	27	23	10	3	1	0	0	0
		Approx. $A_{d(s)}$	1	13	25	26	26	22	11	3	1	0	0	0
	5	True $A_{d(s)}$	1	9	16	15	12	7	3	1	0	0	0	0
		Approx. $A_{d(s)}$	1	9	16	14	12	8	3	1	0	0	0	0
	6	True $A_{d(s)}$	1	6	10	8	4	2	1	0	0	0	0	0
		Approx. $A_{d(s)}$	1	6	10	7	5	2	1	0	0	0	0	0
	7	True $A_{d(s)}$	1	4	5	4	2	0	0	0	0	0	0	0
		Approx. $A_{d(s)}$	1	4	5	3	2	0	0	0	0	0	0	0

Table 3.3. Comparison of True and Approximated $A_{d(s)}$ for BCH code

BCH(15, 7)			d											
			0	5	6	7	8	9	10	15				
s	1-2	True $A_{d(s)}$	Agree Exactly											
		Approx. $A_{d(s)}$												
	3	True $A_{d(s)}$	1	5	5	2	1	1	1	0				
		Approx. $A_{d(s)}$	1	5	6	2	1	1	0	0				
	4-5	True $A_{d(s)}$	Agree Exactly											
		Approx. $A_{d(s)}$												
BCH(31, 16)			d											
			0	7	8	11	12	15	16	19	20	23	24	31
s	1-2	True $A_{d(s)}$	Agree Exactly											
		Approx. $A_{d(s)}$												
	3	True $A_{d(s)}$	1	70	183	1320	1872	2276	1847	424	192	6	1	0
		Approx. $A_{d(s)}$	1	70	183	1321	1871	2275	1848	425	191	6	1	0
	4	True $A_{d(s)}$	1	53	130	800	1072	1058	789	136	56	1	0	0
		Approx. $A_{d(s)}$	1	52	131	802	1069	1056	792	137	55	1	0	0
	5	True $A_{d(s)}$	1	40	90	472	600	472	317	40	16	0	0	0
		Approx. $A_{d(s)}$	1	39	92	475	594	469	323	40	14	0	0	0
	6	True $A_{d(s)}$	1	27	63	279	321	193	124	13	3	0	0	0
		Approx. $A_{d(s)}$	1	28	64	274	320	199	124	11	3	0	0	0
	7	True $A_{d(s)}$	1	18	42	162	168	70	45	6	0	0	0	0
		Approx. $A_{d(s)}$	1	20	43	154	166	79	45	3	1	0	0	0

8	True $A_{d(s)}$	1	12	28	90	84	24	15	2	0	0	0	0
	Approx. $A_{d(s)}$	1	14	29	83	83	30	15	1	0	0	0	0
9	True $A_{d(s)}$	1	8	18	48	40	8	5	0	0	0	0	0
	Approx. $A_{d(s)}$	1	10	19	43	40	10	5	0	0	0	0	0
10	True $A_{d(s)}$	1	5	10	26	20	1	1	0	0	0	0	0
	Approx. $A_{d(s)}$	1	7	12	22	18	3	1	0	0	0	0	0
11	True $A_{d(s)}$	1	2	7	14	8	0	0	0	0	0	0	0
	Approx. $A_{d(s)}$	1	5	7	10	8	1	0	0	0	0	0	0

3.2 Weight Enumerator of Punctured Code

In Section 3.1, we have discussed and shown the effect of code shortening on decoding BER performance. In particular, we show that the weight enumerator of shortened code can be obtained from (3.13) with the use of (3.12). (3.13) can be interpreted as: The ratio of codewords with hamming distance d to the n -tuples with hamming distance d before shortening is approximately the same as the ratio of codewords with hamming distance d to the $(n-s)$ -tuples with hamming distance d after shortening, i.e. $\frac{A_d}{\binom{n}{d}} \approx \frac{A_{d(s)}}{\binom{n-s}{d}}$. Unlike code shortening where information symbols are

nullified, puncturing instead nullifies the parity symbols. In this section, we shall attempt to find the weight enumerator of a punctured code using the afore-mentioned result.

3.2.1 Distance Distribution of Punctured Codes

Consider the case when a single parity bit is being punctured. This parity bit can take the value of either '1' or '0'. Hence the hamming distance can either remain as before or decrease by 1. Fig. 1 shows the true distance and weight distributions of punctured codes from original code: Hamming(15,11,3) with $p=1, 2$ and 3 where p is the number of punctured parity bits. Fig. 1 shows that when p increases by a single bit, hamming distance generally decreases by 1. For example, when $p=1$, the minimum hamming distance decreases from 3 to 2. And when p is increased by 1 again (to 2), the minimum hamming distance remains at 2.

3.2.2 Approximated Weight Enumerator for Punctured Code with Single

Punctured Parity Bit

From Fig. 3.4, similarly to shortened code, it can be observed that the weight distribution of punctured code is binomial-like. Fig. 3.5 shows the true distance and weight distributions of punctured-BCH(p -BCH) (8,7) with $p=7$. It shows that the weight distribution of the punctured code is still binomial-like. Hence we shall apply (3.12) and (3.13), given the hamming distance variation when puncturing is performed (Section (3.2.1)), to find the weight enumerator of punctured code.

Consider the case when the number of punctured parity bit is 1 i.e. $p=1$. Let $A_{d(p)}$ be the number of codewords of hamming distance d for a punctured code. Since puncturing of a *single* parity bit will cause the Hamming distance of the codewords of the original code to either remain the same (if the parity bit happens to be zero) or decrease by 1 (if

the parity bit happens to be 1), the contribution of the codewords to $A_{d(p)}$ of the punctured code can be due to A_d or A_{d+1} of the original code.

Therefore using the “proportion factor” given in (3.12), the weight enumerator for a punctured code can be approximated by

$$A_{d(p)} = \begin{cases} 1, & \text{if } d=0 \\ \text{Round} \left(\frac{\binom{n-1}{d}}{\binom{n}{d}} A_d \right) + \text{Round} \left(\frac{\binom{n-1}{d}}{\binom{n}{d+1}} A_{d+1} \right), & \text{if } d \leq (n-p) \\ 0, & \text{elsewhere} \end{cases} \quad (3.14)$$

Fig. 3.4. True distance and weight distributions of punctured codes from Hamming(15,11,3) with $p=1,2$ and 3

Fig. 3.5. True distance and weight distributions of punctured-Hamming(8,7) with $p=7$

Table 3.4 compares the true and the approximated $A_{d(p)}$ (1) for p-Hamming(6,4). It shows that the true and approximated weight enumerator agree exactly.

Table 3.4. Comparison of True and Approximated $A_{d(p)}$ for p-Hamming(6,4)

Hamming(7,4,3)		d					
		0	2	3	4	6	7
$p=1$	True $A_{d(p)}$	1	3	8	3	1	0
	Approx. $A_{d(p)}$	1	3	8	3	1	0

3.2.3 Approximated Weight Enumerator with Numerous Punctured Parity Bits

The weight enumerator of (3.14) has been obtained based on the assumption that $p=1$. However, when p is larger than 1, (3.14) can be used repeatedly till the number of punctured parity bits p is attained. As an illustration, following the previous example, the distance and weight distribution of punctured code p -Hamming(5, 4) (with $p=2$) can be obtained from the distance and weight distribution of punctured code (6, 4) using (3.14). Hence in order to compute the weight distribution of p -Hamming(5, 4) (with $p=2$) from original code Hamming(7, 4), (3.14) is used twice: first to compute the weight distribution of p -Hamming(6, 4) from Hamming(7, 4) and then the weight distribution of p -Hamming(5, 4) from p -Hamming(6, 4).

3.2.4 Comparison of True and Approximated Weight Enumerators of Punctured Codes

Tables 3.5-3.8 compares the true and approximated weight enumerator of punctured codes from: Hamming(7,4,3) (Table 3.5), Hamming(15,11,3) (Table 3.6), BCH(15,7,5) (Table 3.7) and BCH(15,5,7) (Table 3.8). It has been observed that when p is small ($p=1-2$), the true and approximated weight enumerators agree exactly. However, when p is large (beyond 2), it has been observed that the approximated weight enumerator is not a good estimate.

Table 3.5. Comparison of True and Approximated $A_{d(p)}$ for Hamming(7,4,3)

Hamming(7,4,3)			d							
			0	1	2	3	4	5	6	7
p	1-2	True $A_{d(p)}$ Approx. $A_{d(p)}$	Agree Exactly							

Table 3.6. Comparison of True and Approximated $A_{d(p)}$ for Hamming(15,11,3)

Hamming(15,11,3)			d														
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
p	1-2	True $A_{d(p)}$	Agree Exactly														
		Approx. $A_{d(p)}$															
	3	True $A_{d(p)}$	1	4	34	116	239	392	476	392	239	116	34	4	1	0	0
Approx. $A_{d(p)}$		1	4	34	117	238	391	478	391	238	117	34	4	1	0	0	0

Table 3.7. Comparison of True and Approximated $A_{d(p)}$ BCH(15,7,5)

BCH(15,7,5)			d													
			0	1	2	3	4	5	6	7	8	9	10	11	12	13
p	1	True $A_{d(p)}$	Agree Exactly													
		Approx. $A_{d(p)}$														
	2	True $A_{d(p)}$	1	0	0	2	12	27	22	22	27	12	2	0	0	1
		Approx. $A_{d(p)}$	1	0	0	2	13	26	22	22	26	13	2	0	0	1
	3	True $A_{d(p)}$	1	0	1	4	19	28	22	28	19	4	1	0	1	0
		Approx. $A_{d(p)}$	1	0	0	6	19	26	24	26	19	6	0	0	1	0
	4	True $A_{d(p)}$	1	0	3	6	29	25	25	29	6	3	0	1	0	0
		Approx. $A_{d(p)}$	1	0	2	11	24	27	27	24	11	2	0	1	0	0
	5	True $A_{d(p)}$	1	1	3	16	28	30	28	16	3	1	1	0	0	0
		Approx. $A_{d(p)}$	1	0	5	17	28	29	28	17	5	0	1	0	0	0

Table 3.8. Comparison of True and Approximated $A_{d(p)}$ for BCH(15,5,7)

BCH(15,5,7)		d													
		0	1	2	3	4	5	6	7	8	9	10	11	12	
p	1-2	True $A_{d(p)}$	Agree Exactly												
		Approx. $A_{d(p)}$													
	3	True $A_{d(p)}$	1	0	0	0	1	8	12	8	1	0	0	0	1
		Approx. $A_{d(p)}$	1	0	0	0	1	7	13	7	1	0	0	0	1
	4	True $A_{d(p)}$	1	0	0	0	5	10	10	5	0	0	0	1	0
		Approx. $A_{d(p)}$	1	0	0	0	4	11	11	4	0	0	0	1	0
	5	True $A_{d(p)}$	1	0	0	2	7	12	7	2	0	0	0	0	0
		Approx. $A_{d(p)}$	1	0	0	1	8	12	8	1	0	0	1	0	0
	6	True $A_{d(p)}$	1	0	1	3	11	11	3	1	0	1	0	0	0
		Approx. $A_{d(p)}$	1	0	0	4	11	11	4	0	0	1	0	0	0
	7	True $A_{d(p)}$	1	0	2	8	10	8	2	0	1	0	0	0	0
		Approx. $A_{d(p)}$	1	0	1	8	12	8	1	0	1	0	0	0	0
	8	True $A_{d(p)}$	1	0	6	9	9	6	0	1	0	0	0	0	0
		Approx. $A_{d(p)}$	1	0	4	11	11	4	0	1	0	0	0	0	0

3.3 BER Bounds of Shortened Codes in BER Bounds of Shortened Codes in AWGN Channel

In Section 3.1, we have shown that the approximated weight enumerator for shortened code is a good estimate of the true weight enumerator. However the same is not true for the approximated weight enumerator of punctured code (Section 3.2). Hence for the following sections, we shall only focus on the BER bounds for shortened codes.

3.3.1 BER Bound of Shortened Block Code in AWGN Channel

With code-rate $r_{c(s)} = \frac{k-s}{n-s}$ and weight enumerator of a shortened code given by (3.13), the decoding bit-error probability $P_{b(s)}$ of a shortened code in an AWGN channel can be obtained from (2.7):

$$P_{b(s)} \leq \sum_d \frac{d}{n-s} f_{s,d} A_d P_d(d, r_{c(s)}) \quad (3.15)$$

assuming that the shortened code has the set of Hamming distance $\{d_{ori}\}$ from the original code.

3.3.2 BER Bound for Shortened Product code in AWGN Channel

Consider a 2-D product code (PC) $C^p(n^p, k^p, d^p)$ consisting of horizontal linear block code $C^h(n^h, k^h, d^h)$ whose weight enumerator is A_d^h and vertical linear block code $C^v(n^v, k^v, d^v)$ whose weight enumerator is A_d^v . C^p has minimum Hamming distance $d^p = d^h d^v$, information length $k^p = k^h k^v$, code length $n^p = n^h n^v$ [1] and the number of minimum weight code-words of C^p is given by the product of the number of minimum weight codewords

of the component codes $A_{d^h}^h A_{d^v}^v$ [3]. As the weight distribution of the product code is not easily known, and since the BER is mostly determined by the $d_{\min}$ term at higher E_b/N_o , we may lower-bound the BER by using only the $d_{\min}$ term in (2.7). This bound is known as the *asymptotic* BER bound [3].

Similarly, now consider a 2-D shortened-product code(s-PC) made up of shortened component codes $C_s^h(n^h - s^h, k^h - s^h, d^h)$ and $C_s^v(n^v - s^v, k^v - s^v, d^v)$. Assuming the minimum Hamming distance unchanged, the asymptotic lower BER bound of a 2-D shortened-product code(s-PC) made up of shortened component codes $C_s^h(n^h - s^h, k^h - s^h, d^h)$ and $C_s^v(n^v - s^v, k^v - s^v, d^v)$ is then given by

$$P_{b(s\text{-PC})} \geq \frac{d^h d^v}{(n^h - s^h)(n^v - s^v)} f_{s,d^h}^h f_{s,d^v}^v A_{d^h}^h A_{d^v}^v P_d(d^h d^v, r_c^{(s\text{-PC})}) \quad (3.16)$$

where $r_c^{(s\text{-PC})}$ is the code-rate of the s-PC, and f_{s,d^h}^h and f_{s,d^v}^v scaling factors for the component codes are given by (3.12).

3.3.3 BER Bound for Extended-Shortened Codes

Extended product code (e-PC) refers to a product code made up of extended horizontal and/or vertical component codes. Shortened-extended product code (s-e-PC) refers to a product code made up of extended horizontal and/or vertical component codes which have been shortened.

With code-rate of s-e-PC as $r_c^{(s\text{-e-PC})}$, the asymptotic bounds of s-e-PC in an AWGN channel is given by

$$P_{b(s\text{-e-PC})} \geq \frac{(d^h + 1)(d^v + 1)}{(n^h + 1 - s^h)(n^v + 1 - s^v)} f_{s,(d^h+1)}^{e-h} f_{s,(d^v+1)}^{e-v} A_{(d^h+1)}^{e-h} A_{(d^v+1)}^{e-v} P_d((d^h + 1)(d^v + 1), r_c^{(s\text{-e-PC})}) \quad (3.17)$$

where

$A_{(d^h+1)}^{e-h} = A_{d^h}^h + A_{(d^h+1)}^h$ = number of codewords of weight $d^h + 1$ for the *extended*

horizontal component code (see Section 2.3.1),

$A_{(d^v+1)}^{e-v} = A_{d^v}^v + A_{(d^v+1)}^v$ = number of weight $d^v + 1$ for the *extended* vertical component

code (see Section 2.3.1), and $P_d((d^h + 1)(d^v + 1), r_c^{s-e-PC})$ is given in (2.8).

3.4 BER Bounds of Shortened Codes in Rayleigh Fading

Channel

The asymptotic BER bounds for shortened codes in a Rayleigh fading channel can be easily obtained by replacing the pair-wise error probability of the AWGN channel (2.8) by that of the Rayleigh fading channel (2.16).

Specifically,

- the asymptotic BER bound of shortened block code in a Rayleigh fading channel can be obtained from (3.15) by replacing $P_d(d, r_{c(s)})$ by $\bar{P}_d(d, r_{c(s)})$ and using the d_{min} term only;
- the asymptotic BER bound of s-PC in a Rayleigh fading channel can be obtained from (3.16) by replacing $P_d(d^h d^v, r_c^{(s-PC)})$ by $\bar{P}_d(d^h d^v, r_c^{(s-PC)})$;
- the asymptotic BER bound of s-e-PC in a Rayleigh fading channel can be obtained from (3.17) by replacing $P_d((d^h + 1)(d^v + 1), r_c^{s-e-PC})$ by $\bar{P}_d((d^h + 1)(d^v + 1), r_c^{s-e-PC})$.

3.5 Asymptotic Coding Loss of Shortened Codes

As code shortening typically results in poorer BER, a shortened code typically requires more SNR than an unshortened code to achieve the same decoding bit error probability. This additional SNR, expressed in decibels, required by a shortened code to achieve a specific bit error probability as the unshortened code is termed *coding loss*. The *asymptotic coding loss* is obtained when the asymptotic BER bounds of the unshortened code and shortened code are used to find the coding loss. In this section, we derive the generic coding loss expressions when a code is shortened.

The asymptotic coding loss in an AWGN channel between a shortened code and an un-shortened can be estimated by

- (1) using the asymptotic BER bound and,
- (2) bounding $Q(x)$ in (2.8) by $\frac{1}{2}\exp(-x^2/2)$ [23].

Using the scaling factor $f_{s,d}$ from (3.12) and with d_{min} of shortened code unchanged, the coding loss L_{awgn} in an AWGN channel is given by

$$L_{awgn} \text{ (dB)} = 10 \log_{10} \left(\frac{r_c}{r_{c(s)}} + \frac{\ln \left(\frac{n}{n-s} \right) + \ln f_{s,d_{min}}}{d_{min} r_{c(s)} \frac{E_b}{N_o}} \right). \quad (3.18)$$

Note that if there is truly a coding loss after code shortening, (3.18) will give positive values, otherwise the $\text{Log}(\cdot)$ term will give negative values, which imply coding gain instead of coding loss after code shortening. Assuming large E_b/N_o , (3.18)

becomes $10 \log_{10} \left(\frac{r_c}{r_{c(s)}} \right)$, which is always positive because r_c is always larger than $r_{c(s)}$

and hence a coding loss.

In Rayleigh fading channel, the coding loss is similarly estimated by

(1) using the asymptotic BER bound and,

$$(2) \text{ approximating } \bar{P}_d(d, r_c) \text{ by } \bar{P}_d(d, r_c) \leq \frac{1}{2} \left[\frac{1}{1 + r_c \frac{E_b}{N_o}} \right]^d \quad [26].$$

Using scaling factor $f_{s,d}$ from (3.12) and with d_{min} of shortened code unchanged, the coding loss $L_{rayleigh}$ in a Rayleigh fading channel is then given by

$$L_{rayleigh} \text{ (dB)} = 10 \log_{10} \left(\frac{G-1}{r_{c(s)} \frac{E_b}{N_o}} + \frac{Gr_c}{r_{c(s)}} \right) \quad (3.19)$$

where

$$G = \left(\frac{n}{n-s} f_{s,d_{min}} \right)^{\frac{1}{d_{min}}} \quad (3.20)$$

Similarly, (3.19) will give positive values for coding loss and negative values for coding

gain. Assuming large E_b/N_o , (3.19) becomes $10 \log_{10} \left(\frac{Gr_c}{r_{c(s)}} \right)$ with $\frac{n}{n-s} > 1$, $0 < f_{s,d_{min}} < 1$

and $d_{min}^{-1} < 1$. So depending on the code parameters (r_c , n , d_{min}) and the amount of shortening (s), coding gain or coding loss may results.

3.6 Simulations and Discussions

In this section, we will verify the tightness of the BER bounds developed in the previous sections for shortened block codes, shortened product codes and shortened-extended product codes using computer simulations and other benchmarks. In the following discussions, shortened BCH code is abbreviated as s-BCH and shortened Hamming code as s-Hamming.

3.6.1 Results for BER Bounds

Fig. 3.6 shows the simulated BER and the BER union bound values for s-Hamming(6, 3) and s-BCH(14, 6) with 1 nullified bit, i.e. $s = 1$. The simulated BER is obtained by Monte Carlo simulations of maximum-likelihood soft decoding using all possible codewords, i.e. brute-force decoding. The BER bound is obtained from (3.15). The results in the figure show that the bound (3.15) obtained using the weight enumerator scaling factor $f_{s,d}$ in (3.12) is tight and matches the simulated BER values very closely at mid-to-high E_b/N_o values. At low E_b/N_o values, the bound is looser and lies above the simulated BER line because union bound is by nature an overbound (upper bound).

Fig. 3.6. Proposed BER bounds and simulated BER values for Hamming(7, 4) and BCH(15, 7) shortened by 1 bit ($s = 1$) in AWGN channel

Fig. 3.7 shows the simulated BER and its BER union bound for BCH(31, 16) with more shortened symbols ($s = 8$ and $s = 10$). The results again show that, notwithstanding the larger s , the bound obtained in (3.15) is tight for $BER < 10^{-3}$.

Fig. 3.7. Proposed BER bounds and simulated BER values for BCH(31,16) shortened by $s = 8$ and $s = 10$ in AWGN channel

Fig. 3.8 shows the simulated BER and asymptotic lower bound in AWGN and Rayleigh fading channels for the shortened-extended product code $s\text{-e-PC}(11,6,4)^2$, which is obtained from the extended product code $e\text{-PC}(16,11,4)^2$ by shortening the first 5 bits in all rows and columns ($s^h = s^v = 5$). As this code is large, its simulated BER results are obtained using Pyndiah's decoding algorithm (described in Sections 2.5 and 2.6) with 6 test positions (i.e. 64 test patterns) and 4 decoding iterations. More test positions and decoding iterations do not improve the performance further. The results again show that the proposed tight asymptotic bounds (3.17) match well with the true BER performance.

Fig. 3.8. Proposed asymptotic BER bounds and simulated BER values for s-e-PC(11,6,4)² shortened by $s^h=s^v=5$ in AWGN and Rayleigh fading channels

Chapter 4

Ordered Shortening

In this chapter, we propose a novel code shortening scheme, called *Ordered Shortening*, which carefully selects specific information symbol positions to shorten, instead of shortening a block of consecutive information symbol positions as in the conventional code shortening scheme, in order to obtain better code properties and decoding performance. Before we go into details, we shall explain the motivation behind this work.

4.1 Motivation

Table 4.1 compares the d_{min} values of conventional shortened Hamming and BCH codes with the theoretical upper bound values of d_{min} obtained from [27][29][29]. The number of nullified information symbol positions is given by s . It is observed that the d_{min} values of these shortened codes are not optimal, in the sense that they do not meet the theoretical upper bound values. For example, when a Hamming(15, 11) code is shortened by 7 information symbols using the conventional shortening scheme as described in Section 2.3.2, the shortened Hamming(8,4) code has a d_{min} of 3 while the upper bound states that the best possible (8,4) code is able to have a d_{min} value of 4. This observation

motivates us to investigate other ways of code shortening with a view to maximize the d_{min} value or minimize the corresponding $A_{d_{min}}$ value of the shortened codes.

Table 4.1. Comparison of d_{min} of conventional shortened codes and theoretical upper-bound values of d_{min}

Code Type	(n, k) of Original Code	s	d_{min}	Upper Bound of d_{min}
Hamming	(15, 11)	7	3	4
	(31, 26)	15	3	4
	(63, 57)	31	3	4
BCH	(15, 7)	3	5	6
	(15, 5)	3	7	8
	(31, 21)	7	5	6
	(31, 11)	4	11	12
	(31, 6)	3	15	16

4.2 Ordered Shortening

4.2.1 Shortening from the Parity Check Matrix Point of View

A systematic codeword e has the form $e = [u, m]$ where $u = [u_1, u_2, \dots, u_{n-k}]$ and $m = [m_1, m_2, \dots, m_k]$ constitute the parity-check portion and information message of the codeword respectively. Given the parity check matrix $[H] = [I_{n-k} \ P^T]$ where $[I_{n-k}] = [i_1, i_2,$

$\dots, i_{n-k}]$ and $[\mathbf{P}^T] = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_i, \dots, \mathbf{p}_k]$, the codeword $\mathbf{e}$ of a code satisfies the condition $\mathbf{e} \cdot [\mathbf{H}]^T = \mathbf{0}$ (see Section 2.1.3). That is,

$$u_1 \cdot i_1 + u_2 \cdot i_2 + \dots + u_{n-k} \cdot i_{n-k} + m_1 \mathbf{p}_1 + m_2 \mathbf{p}_2 + \dots + m_i \mathbf{p}_i + \dots + m_k \mathbf{p}_k = \mathbf{0}. \quad (4.1)$$

When a particular information symbol m_i is shortened, its value is permanently put to zero (nullified), hence its contribution $m_i \mathbf{p}_i$ in (4.1) will be removed. This is equivalent to taking away the corresponding column $\mathbf{p}_i$ from the matrix $[\mathbf{P}^T]$ or $[\mathbf{H}]$. Hence, there is a one-to-one correspondence between the respective m_i and $\mathbf{p}_i$ as shown by the arrow in Fig. (4.1).

Fig. 4.1. Correspondence between m_i and $\mathbf{p}_i$

4.2.2 Principle of Ordered Shortening

In order to obtain shortened codes with large d_{min} value, we propose to shorten a code by removing codewords with the smallest Hamming distances first. As explained in Section 2.1.3, the minimum distance of a systematic linear block code is equal to the smallest number of columns from $[\mathbf{H}] = [\mathbf{I}_{n-k} \ \mathbf{P}^T]$ that sum to $\mathbf{0}$. Since $[\mathbf{I}_{n-k}]$ is an identity matrix, any number of its columns will not sum to $\mathbf{0}$. Given that $[\mathbf{I}_{n-k}] = [i_1, i_2, \dots, i_{n-k}]$, the set $\{I_{basis}\} = \{i_1, i_2, \dots, i_{n-k}\}$ forms the basis for the column-space of the parity check

matrix $[\mathbf{H}] = [\mathbf{I}_{n-k} \mathbf{P}^T]$, which implies that any column (or sum of columns) in $[\mathbf{P}^T] = [\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_i, \dots, \mathbf{p}_k]$ can be summed with the basis vectors from $\{I_{basis}\}$ to give the zero vector $\mathbf{0}$.

Suppose we select certain n_c columns from $[\mathbf{P}^T]$. Define the indicator vector $\mathbf{z} = [z_1, z_2, \dots, z_i, \dots, z_k]$ such that

$$z_i = \begin{cases} 1, & \text{if } \mathbf{p}_i \text{ is selected} \\ 0, & \text{if } \mathbf{p}_i \text{ is not selected} \end{cases} \quad (4.2)$$

For example, $\mathbf{z} = [1 \ 1 \ 0 \ 0]$ corresponds to the selection of $\mathbf{p}_1$ and $\mathbf{p}_2$. By selecting these n_c columns from $[\mathbf{P}^T]$, we are actually selecting a particular systematic codeword $\mathbf{e}$ with the information message $\mathbf{m} = \mathbf{z}$. That is, the selection of a codeword with information weight n_c . The information weight is defined as the number of ones in the information message $\mathbf{m}$ of a codeword.

Let $\mathbf{v}$ be the vector formed by the modulo-2 summation of these selected n_c distinct column vectors $\mathbf{p}_i$. Since $\mathbf{v}$ can be summed with the basis vectors from $\{I_{basis}\}$ to give the zero vector $\mathbf{0}$, the weight w_{n_c} of $\mathbf{v}$ will then indicate the number of basis vectors needed to give the zero vector $\mathbf{0}$. w_{n_c} is also the parity check weight of the codeword i.e. the number of ones in the parity check sequence of the codeword. The Hamming distance d of the codeword with information message $\mathbf{m} = \mathbf{z}$ is hence given by $d = n_c + w_{n_c}$.

If d_{min} is the minimum Hamming distance of the original (n, k) code, the codewords with weight d_{min} will hence give a weight of $w_{n_c} = d_{min} - n_c$ for $\mathbf{v}$, the sum of n_c columns from $[\mathbf{P}^T]$. This property enables us to identify the codewords with weight d_{min} to remove by shortening the associated information symbol positions. This process is explained in the algorithm below, followed by an example.

4.3 Algorithm

Given that $[\mathbf{P}^T] = [p_1, p_2, \dots, p_i, \dots, p_k]$, the set $\{\mathbf{v}\}_{n_c}$ is the set of $\mathbf{v}$ formed by all the possible modulo-2 summation of n_c distinct p_i identified by $\mathbf{z}$ (4.2), i.e.

$$\begin{aligned} \text{if } n_c = 1, \quad \{\mathbf{v}\}_{n_c} &= \{p_i\}_{\text{all } i} \\ \text{if } n_c = 2, \quad \{\mathbf{v}\}_{n_c} &= \{p_i \oplus p_j\}_{\text{all } i \neq j} \\ \text{if } n_c = 3, \quad \{\mathbf{v}\}_{n_c} &= \{p_i \oplus p_j \oplus p_l\}_{\text{all } i \neq j \neq l} \\ &\vdots \end{aligned} \tag{4.3}$$

for $1 \leq i, j, l \leq k$ and $1 \leq n_c \leq d_{\min}$.

The steps of the *ordered shortening* scheme are:

1. Starting with $n_c = 1$, form all possible $\mathbf{v}$ vectors using (4.3).
2. The weight w_{n_c} of each $\mathbf{v}$ is computed. For example, if $\mathbf{v} = \begin{pmatrix} 0 \\ 1 \\ 1 \end{pmatrix}$, its weight $w_{n_c} = 2$.
3. Pick all $\mathbf{z}$ with $w_{n_c} = d_{\min} - n_c$. They correspond to the set of minimum-Hamming-distance codewords denoted by $\{\mathbf{z}\}_{n_c}$.
4. Record the number of times that each $z_i = 1$ has occurred in the set of $\{\mathbf{z}\}_{n_c}$ from step 3.
5. Shorten (nullify) the information symbol position corresponding to the highest frequency of $z_i = 1$ occurrence.
6. Remove those $\mathbf{z}$ vectors identified in step 5 from $\{\mathbf{z}\}_{n_c}$.
7. Repeat steps 5 and 6 till all the $\mathbf{z}$ vectors in $\{\mathbf{z}\}_{n_c}$ are removed, or until the number of shortened (or nullified) positions s has been met, whichever occurs earlier.

8. Increment n_c and repeat step 1-7 until $n_c = d_{min}$.

Note that if $n_c > d_{min}$, all the minimum Hamming distance codewords have already been removed.

4.3.1 Example of Ordered Shortening

We shall now illustrate the above shortening algorithm with a simple example. A small code, Hamming(15, 11) with $d_{min} = 3$, is used in this example. Its parity check matrix is given by

$$\begin{aligned} [\mathbf{H}] &= [\mathbf{I}_4 \quad \mathbf{P}_{11}^T] \\ &= \begin{bmatrix} 1 & 0 & 0 & 0 & : & 1 & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & : & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & : & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & : & 0 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \end{aligned} \quad (4.4)$$

With $n_c = 1$, $\{\mathbf{v}\}_{n_c=1} = \{\mathbf{p}_1, \mathbf{p}_2, \dots, \mathbf{p}_{11}\}$. The set of information sequences $\{\mathbf{z}\}_{n_c}$ with information weight $n_c = 1$ from the systematic codewords are

$$\{\mathbf{z}\}_{n_c=1} = \left\{ \begin{array}{l} \mathbf{z}_1 = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_2 = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_3 = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_4 = [0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_5 = [0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_6 = [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_7 = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_8 = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_9 = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_{10} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0], \\ \mathbf{z}_{11} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0] \end{array} \right\}. \quad (4.5)$$

For ease of identification, we have labeled these information sequences from z_1 to z_{11} .

The parity-check weights of the codewords identified by $\{z\}_{n_c}$ are given by the weights of the respective columns from $[P_{11}^T]$ to be 2, 2, 2, 3, 2, 2, 3, 3, 4, 3, and 2. The z patterns from (4.5) that satisfy the criterion $w_{n_c} = d_{\min} - n_c = 2$ are

$$\{z|w_{n_c}=2\}_{n_c=1} = \left\{ \begin{array}{l} z_1 = [1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ z_2 = [0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ z_3 = [0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ z_5 = [0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ z_6 = [0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0], \\ z_{11} = [0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1] \end{array} \right\}. \quad (4.6)$$

These are the information sequences of the identified minimum Hamming distance codewords. Hence if the desired $s = 6$, z_1, z_2, z_3, z_5, z_6 and z_{11} are to be removed from $\{z\}_{n_c=1}$. Therefore, information symbol positions 1, 2, 3, 5, 6 and 11 are to be shortened and the parity check columns p_1, p_2, p_3, p_5, p_6 and p_{11} will be removed from the parity check matrix. If $s = 7$, we have to increment n_c and repeat the above using the shortened parity check matrix $[H_r] = [I_4 \ p_4 \ p_7 \ p_8 \ p_9 \ p_{10}]$ which consists of the original identity matrix I_4 and the remaining p_i .

With $n_c = 2$, the z -patterns that meet the criterion $w_{n_c} = d_{\min} - n_c = 1$ are

$$\{z|w_{n_c}=1\}_{n_c=2} = \left\{ \begin{array}{l} z_1 = [0 \ 0 \ 1 \ 1 \ 0], \\ z_2 = [0 \ 1 \ 0 \ 1 \ 0], \\ z_3 = [1 \ 0 \ 0 \ 1 \ 0] \end{array} \right\} \quad (4.7)$$

with the columns corresponding to p_4, p_7, p_8, p_9 and p_{10} respectively. For convenience, we

have labeled them as z_1 to z_3 . These are the information sequences of minimum Hamming distance codewords with $n_c = 2$. We may remove these minimum Hamming distance codewords by shortening the corresponding information bit 4, 7, 8, 9 or 10. Clearly the information bit to shorten is the one associated with the most number of minimum Hamming distance codewords i.e. bit 9 in this case. Combined with the previous results, the information symbol positions to be shortened for $s = 7$ are therefore 1, 2, 3, 5, 6, 11 and 9.

In the above steps are repeated for $n_c = d_{min} = 3$, no z -patterns with $w_{n_c} = d_{min} - n_c = 0$ are found. Hence, performing *ordered shortening* with $s = 7$ on a Hamming(15, 11) results in a shortened code with increased $d_{min} = 4$.

Note that the search space of the proposed *ordered shortening* algorithm is not more than $\sum_{n_c=1}^{d_{min}} \binom{k}{n_c}$ or $\sum_{n_c=1}^k \binom{k}{n_c}$ whichever is smaller. Hence the algorithm is generally much more efficient than a brute-force search over all combinations of k information bits.

4.4 Results

4.4.1 Ordered-Shortened Hamming and BCH Codes

Table 4.2 shows the set of information symbol positions $\{i_s\}$ to be shortened until an increase in d_{min} is achieved for Hamming and BCH codes. In the table, the symbols n , k and d_{min} are parameters of the original un-shortened code. The increased minimum Hamming distance value is $d_{min}+1$. From the table, we can see that *ordered-shortened* codes reach an increase in d_{min} faster than the conventional-shortened codes. Moreover, the conventional shortening scheme does not produce an increase in d_{min} at all for some codes such as Hamming(15, 11), Hamming(31, 26), BCH(15, 7), BCH(15, 5) and BCH(31, 11).

Table 4.2. Information symbol positions to shorten to achieve an increased d_{min} for Hamming and BCH codes with $n = 15$ and 31

Table 4.2a. Information symbol positions to shorten to achieve an increased d_{min} for Hamming codes with $n = 15$ and 31

Hamming				
Original code			$\{i_s\}$	
n	k	d_{min}	Ordered-shortened code	Conventional-shortened code
15	11	3	{1, 2, 3, 5, 6, 9, 11}	-*
31	26	3	{1, 2, 3, 6, 14, 15, 16, 17, 25, 26, 10, 12, 19, 20, 22}	-*

Table 4.2b. Information symbol positions to shorten to achieve an increased d_{min} for BCH codes with $n = 15$ and 31

BCH				
Original code			$\{i_s\}$	
n	k	d_{min}	Ordered-shortened code	Conventional-shortened code
15	7	5	{1, 4, 5, 6, 7}	-*
	5	7	{1, 2, 5}	-*
31	21	5	{4, 5, 14, 15, 16, 1, 8, 10, 19, 20, 21}	{1, 2, ... , 17}
	16	7	{6, 8, 9, 10, 11, 12, 1, 2, 3, 4, 16}	{1, 2, ... , 11}
	11	11	{1, 2, 10, 11, 4}	-*

* No increase in d_{min}

4.4.2 Comparison of Minimum Hamming Distance with Theoretical Upper Bound

Bound

Fig. 4.2. Comparison of d_{min} between conventional shortening, *ordered* shortening and the best $(31-s, 26-s)$ possible code. Original code= Hamming(31,26,3).

Fig. 4.2 shows the minimum Hamming distance d_{min} of Hamming(31, 26, 3) shortened by the conventional shortening and the proposed *ordered shortening* schemes, alongside the corresponding theoretical upper bound values of d_{min} obtained from [27][29][29] for the best possible $(31-s, 26-s)$ code. It is clear that the *ordered shortening* scheme produces an increase in d_{min} of 1 after shortening 15 information symbols (i.e. $s = 15$), while the conventional shortening scheme does not, even after shortening all information symbols. Furthermore, the d_{min} of *ordered-shortened* code meets the theoretical upper bound.

4.4.3 Comparison of Minimum Weight Enumerator

Next we study the effect of *ordered* shortening on the weight enumerator A_d before an increase in d_{min} is achieved. Fig. 4.3a and 4.3b show the weight enumerator A_d of s-Hamming(18, 13) and s-PC(18, 13)² respectively, obtained from Hamming(31, 26) with $s = 13$. Note that only the smallest few (hence most significant) values of d are considered here. According to [9], the smallest three values of d of a product code made up of non-extended Hamming codes are 9, 12 and 15, and the corresponding weight enumerators are

$$A_d = \begin{cases} A_3^h \cdot A_3^v, & \text{if } d = 9 \\ A_3^h \cdot A_4^v + A_4^h \cdot A_3^v, & \text{if } d = 12 \\ A_3^h \cdot A_5^v + A_5^h \cdot A_3^v, & \text{if } d = 15 \end{cases} \quad (4.8)$$

where the superscripts h and v refer to the horizontal and vertical component codes of the product code respectively. Hence, the weight enumerator values A_9 , A_{12} and A_{15} of Fig. 4.3b are computed using (4.8) based on the A_3 , A_4 and A_5 values from Fig. 4.3a.

Fig. 4.3. Comparison of A_d between *ordered* and *conventional* shortening for (a) s-Hamming(18, 13) and (b) s-PC(18, 13)²

Fig. 4.3a shows that the proposed *ordered shortening* scheme produces s-Hamming(18, 13) with smaller A_d values than *conventional shortening* for $d = 3$ and 5. This is also true for s-PC(18, 13)² as shown in Fig. 4.3b, for all values of $d = 9, 12$ and 15. This shows that besides producing increased d_{min} faster (with smaller s), the *ordered shortening* scheme is also able to produce smaller A_d values for the smaller d terms. Note also that although the A_4 value in s-Hamming(18,13) of *ordered shortening* is larger, the A_{12} value in s-PC(18,13)² of *ordered shortening* is smaller. This is because the A_{12} value of s-PC(18,13)² is contributed by both the A_3 and A_4 values of s-Hamming(18,13).

Since the proposed *ordered shortening* scheme is capable of producing larger d_{min} or smaller A_d values, the BER performance of *ordered-shortened* code is expected to be

better than that of the conventional-shortened code. We shall verify this in the next section.

4.4.4 Comparison of Simulated BER

Fig. 4.4 shows the simulated BER and the asymptotic BER bound (obtained from (2.7) using only the d_{min} term) for s-PC(17,12)² with *ordered* and conventional shortening in AWGN channel. The simulation results are obtained using Pyndiah's algorithm with 6 test positions and 4 iterations. Both the bound and simulated results show that the *ordered-shortened* code has lower BER than the *conventional-shortened* code. Specifically, at a BER of 10^{-6} , the *ordered-shortened* code outperforms the *conventional-shortened* code by about 0.8dB.

Fig. 4.5 shows the simulated BER and the asymptotic BER bound for s-PC (33,27)² with *ordered* and conventional shortening in AWGN channel. At a BER of 10^{-6} , the *ordered-shortened* code's outperforms the *conventional-shortened* code by about 0.6 dB.

Fig. 4.4. Simulated and analytical BER performance of s-PC(17, 12)² with *ordered* and *conventional* shortening in AWGN channel

Fig. 4.5. Simulated and analytical BER performance of s-PC(33, 27)² with *ordered* and *conventional* shortening in AWGN channel

Fig. 4.6 shows the simulated BER and asymptotic BER bound (obtained from (2.9) using only the $d_{min} = 9$ term) for s-PC $(17, 12)^2$ with *ordered* and conventional shortening in Rayleigh fading channel. Better BER performance is again observed for the *ordered-shortened* code. At BER of 10^{-6} , the difference is about 1 dB.

Fig. 4.6. Simulated and analytical BER performance of s-PC $(17, 12)^2$ with *ordered* and conventional shortening in Rayleigh fading channel

Note that for Fig. 4.4, 4.5 and 4.6, the d_{min} (=9) of the conventional and *ordered* shortened codes are the same. The performance gain is hence mainly due to a reduced $A_{d_{min}}$.

Fig. 4.7 shows the simulated BER and the asymptotic BER bound for s-PC $(20, 10)^2$ with *ordered* and conventional shortening in AWGN channel. It illustrates the case

where the d_{min} of the *ordered-shortened* codes is larger than the d_{min} of the conventional-shortened codes for the same amount of shortening ($s=11$). The conventional-shortened code has $d_{min} = 5 \times 5 = 25$ while the *ordered-shortened* code has $d_{min} = 6 \times 6 = 36$. The bigger d_{min} has evidently resulted in a steeper BER slope for the *ordered-shortened* code and hence a diverging BER gap compared to the conventional shortened code. This makes the *ordered-shortened* product code particularly attractive for systems requiring very low BER.

Fig. 4.7. Simulated and analytical BER performance of $s\text{-}PC(20, 10)^2$ with *ordered* and *conventional* shortening in AWGN channel

Chapter 5

Turbo Decoding of Product Code in Pulsed Jamming Channels

In this chapter, we extend Pyndiah's turbo decoding algorithm for product codes (PC) to the pulsed jamming channels and benchmark its performance with the BER bounds.

5.1 Pulsed Jamming Noise Model

We consider an interferer who jams a fraction ρ ($0 < \rho < 1$) of the symbols in a codeword. The fraction of coded symbols not jammed is equal to $(1 - \rho)$. ρ is also known as the duty cycle of the jammer. We model the jamming signal as white Gaussian noise whose one-sided power spectral density (PSD) is N_j/ρ when the jammer is on, and 0 when the jammer is off [30]. The average jammer's PSD (one-sided) is then N_j .

Hence, the one-sided PSD of the jamming noise is

$$N_J = \begin{cases} \frac{N_J}{\rho}, & \text{with probability } \rho, \\ 0, & \text{with probability } (1-\rho) \end{cases} \quad (5.1)$$

and the one-sided PSD of the total noise N_T for a particular symbol is then given by

$$N_T = \begin{cases} \frac{N_J}{\rho} + N_0, & \text{if jammer is on} \\ N_0, & \text{if jammer is off} \end{cases} \quad (5.2)$$

where N_0 is the one-sided gaussian noise spectral density.

We further assume that the probability that a symbol is jammed is independent of the rest of the symbols, i.e. the jamming effect is memoryless across the coded symbols.

5.2 BER Bounds for Pulsed Jamming Channels

In this section, we shall derive the BER union bounds for a pulsed jamming channel for use as reference for the simulated BER results to be obtained later on. In the derivation, we assume that the decoder has the knowledge of the jamming state information (JSI) which includes N_J and ρ , and the channel state information (CSI) which includes the fading coefficients and N_0 .

5.2.1 Pulsed Jamming AWGN Channel with JSI

The upper bound on the BER of a linear code in AWGN channel is shown in (2.7). To find the corresponding BER bound for a pulsed jamming channel with AWGN channel noise, we need to first condition the pairwise error probability (2.8) on the

number of jammed symbols n_J out of d symbols in a codeword and then average this conditioned pairwise error probability over the distribution of n_J .

Based on the assumption that the probability that a symbol is jammed is memoryless, we can model n_J as having a binomial distribution with probability of success, ρ (probability of a symbol being jammed) and the maximum value of n_J , d (code Hamming distance).

Therefore the pdf of n_J is given by

$$P(n_J) = \sum_{n_J=0}^d \binom{d}{n_J} \rho^{n_J} (1-\rho)^{d-n_J} \quad (5.3)$$

The pair wise error probability in a pulsed jamming AWGN channel is then given by

$$\begin{aligned} P_d(d, r_c) &= P_h(d, r_c | n_J) \cdot P(n_J) \\ &= \sum_{n_J=0}^d \binom{d}{n_J} \rho^{n_J} (1-\rho)^{d-n_J} \cdot P_d(d, r_c | n_J) \end{aligned} \quad (5.4)$$

where $P_d(d, r_c | n_J)$ is the pairwise error probability conditional on n_J of the d symbols being jammed.

Given that n_J out of d symbols in a codeword are jammed, we can separate these d symbols into n_J jammed and $(d - n_J)$ unjammed symbols with different noise PSD as given in (5.2) respectively and modify (2.8) accordingly to give

$$P_d(d, r_c | n_J) = Q \left(\sqrt{2r_c E_b \left(\frac{d - n_J}{N_0} + \frac{n_J}{N_0 + N_J / \rho} \right)} \right) \quad (5.5)$$

Using (5.4) in (2.7), the soft decoding bit error probability P_b in a pulsed jamming AWGN channel is then upper-bounded by

$$\begin{aligned} P_b &\leq \sum_d \frac{d}{n} A_d \cdot P_d(d, r_c | n_J) \cdot P(n_J) \\ &\leq \sum_d \frac{d}{n} A_d \sum_{n_J=0}^d \binom{d}{n_J} \rho^{n_J} (1-\rho)^{d-n_J} \cdot P_d(d, r_c | n_J) \end{aligned} \quad (5.6)$$

where $P_d(d, r_c | n_J)$ is given in (5.5).

5.2.2 Pulsed Jamming Rayleigh Fading Channel with JSI and CSI

The approach to finding the BER union bound for a pulsed jamming Rayleigh fading channel with JSI and CSI is similar to that in Section (5.2.1), except that we have to condition the pairwise error probability $\bar{P}_d(d, r_c | \mathbf{a}, n_J)$ on 2 random variables now:

- (1) $\mathbf{a} = [a_1, \dots, a_l, \dots, a_d]$, the fading coefficient vector for $\mathbf{r} = [r_1, \dots, r_l, \dots, r_d]$ and,
- (2) n_J , the number of jammed symbols out of d symbols.

Subsequently, we will also need to average $\bar{P}_d(d, r_c | \mathbf{a}, n_J)$ over the pdfs of $\mathbf{a}$ and n_J .

Following the derivation of (5.5), we separate the d symbols from (2.10) into jammed and unjammed symbols where the n_J jammed and $(d - n_J)$ unjammed symbols have noise PSD as given in (5.2) to obtain

$$P_d(d, r_c | n_J, \mathbf{a}) = Q \left(\sqrt{2r_c \left(\left(\frac{E_b}{N_J / \rho + N_0} \right) \sum_{i=1}^{n_J} a_i^2 + \frac{E_b}{N_0} \sum_{i=n_J+1}^d a_i^2 \right)} \right) \quad (5.7)$$

Noting (2.15), $P_d(d, r_c | \mathbf{a}, n_J)$ can be expressed as

$$P_d(d, r_c | n_j, \mathbf{a}) = \frac{1}{\pi} \int_0^{\pi/2} e^{-\frac{\left(\frac{r_c E_b}{N_J / \rho + N_0} \sum_{i=1}^{n_J} a_i^2 + \frac{r_c E_b}{N_0} \sum_{i=n_J+1}^d a_i^2 \right)}{\sin^2 \phi}} d\phi \quad (5.8)$$

To compute the average error probability $\bar{P}_d(d, r_c | n_j)$, $P_d(d, r_c | n_j, \mathbf{a})$ is averaged over $\mathbf{a}$ to give

$$\begin{aligned} \bar{P}_d(d, r_c | n_j) &= \int_{\mathbf{a}} p_{\mathbf{a}}(\mathbf{a}) P_d(d, r_c | n_j, \mathbf{a}) d\mathbf{a} \\ &= \int_{a_1} \cdots \int_{a_d} P_d(d, r_c | n_j, (a_1, a_2, \dots, a_d)) p_{\mathbf{a}}(a_1, a_2, \dots, a_d) da_1 \cdots da_d \end{aligned} \quad (5.9)$$

where $p_{\mathbf{a}}(a_1, a_2, \dots, a_d)$ is given in (2.13). Assuming independent Rayleigh fading, the d -dimensional integral in (5.9) reduces to a product of integrals over each a_i .

Substituting (5.8) and (2.13) into (5.9), it can be shown that (see Appendix C for the derivation)

$$\bar{P}_d(d, r_c | n_j) = \frac{1}{\pi} \int_0^{\pi/2} \left(\frac{\sin^2 \phi}{\frac{r_c E_b}{N_J / \rho + N_0} + \sin^2 \phi} \right)^{n_J} \left(\frac{\sin^2 \phi}{\frac{r_c E_b}{N_0} + \sin^2 \phi} \right)^{d-n_J} d\phi \quad (5.10)$$

The soft decoding bit error probability P_b in a pulsed jamming Rayleigh fading channel is then upper-bounded by

$$\begin{aligned} P_b &\leq \sum_d \frac{d}{n} A_d \bar{P}_d(d, r_c | n_j) P(n_j) \\ &\leq \sum_d \frac{d}{n} A_d \sum_{n_j=0}^d \binom{d}{n_j} \rho^{n_j} (1-\rho)^{d-n_j} \bar{P}_d(d, r_c | n_j) \end{aligned} \quad (5.11)$$

5.3 Pyndiah's SISO Decoder for Pulsed Jamming Channel

5.3.1 Decoding in Pulsed Jamming AWGN Channel

5.3.1.1 Case I: Decoding with JSI

In addition to the jammer's PSD N_J and jamming factor ρ , jammer state information (JSI) includes the knowledge of the number of jammed symbols and in particular, which symbols are jammed. In practical situations, estimation of the JSI is required to predict whether a symbol has been jammed. This, however, will not be discussed in this thesis. We will mainly derive 2 new results:

- Channel reliability values with JSI
- Decoder soft output with JSI

Channel Reliability Values:

The channel reliability value generated by the demodulator from $\mathbf{r}$ and jammer state information $\mathbf{b} = [b_1, \dots, b_j, \dots, b_n]$ using the log-likelihood ratio (LLR) of decision r_j is given by

$$\Lambda(r_j) = \ln \left(\frac{\Pr\{x_j = +1 | r_j, b_j\}}{\Pr\{x_j = -1 | r_j, b_j\}} \right) \quad (5.12)$$

where b_j is defined as

$$b_j = \begin{cases} 0, & \text{if bit } j \text{ is unjammed} \\ 1, & \text{if bit } j \text{ is jammed} \end{cases} \quad (5.13)$$

With the total PSD given in (5.2), the pdf of each pulse-jammed-AWGN-channel noise sample n_j is given by

$$p(n_j|b_j) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left[-\frac{n_j^2}{2\sigma_n^2}\right], & \text{if } b_j = 0 \\ \frac{1}{\sqrt{2\pi\left(\sigma_n^2 + \sigma_J^2/\rho\right)}} \exp\left[-\frac{n_j^2}{2\left(\sigma_n^2 + \sigma_J^2/\rho\right)}\right], & \text{if } b_j = 1 \end{cases} \quad (5.14)$$

where σ_n^2 is the AWGN noise's variance and σ_J^2/ρ is the jammer noise's variance.

Given that x_j is transmitted, the pdf of a received symbol r_j from a pulsed jamming AWGN channel is

$$p(r_j|x_j, b_j) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma_n^2}} \exp\left[-\frac{(r_j - x_j)^2}{2\sigma_n^2}\right], & \text{if } b_j = 0 \\ \frac{1}{\sqrt{2\pi\left(\sigma_n^2 + \sigma_J^2/\rho\right)}} \exp\left[-\frac{(r_j - x_j)^2}{2\left(\sigma_n^2 + \sigma_J^2/\rho\right)}\right], & \text{if } b_j = 1 \end{cases} \quad (5.15)$$

Using Bayes' rule, the reliability value in (5.12) can be shown to be

$$\Lambda(r_j) = \begin{cases} \frac{2}{\sigma_n^2} r_j, & \text{if } b_j = 0 \\ \frac{2}{\sigma_n^2 + \sigma_J^2/\rho} r_j, & \text{if } b_j = 1 \end{cases} \quad (5.16)$$

The absolute value of this channel reliability value, $|\Lambda(r_j)|$, is used to identify the least reliable received bits for forming the test patterns in the Pyndiah/Chase algorithm (see Section 2.5.1.3).

Soft Output from Decoder:

The reliability of decision d_j at the output of the channel decoder defined using log-likelihood ratio (LLR) is given by

$$\Lambda(d_j) = \ln \left(\frac{\sum_{c^i \in \{s_j^{+1}\}} p(\mathbf{r} | \mathbf{x} = c^i, n_j)}{\sum_{c^i \in \{s_j^{-1}\}} p(\mathbf{r} | \mathbf{x} = c^i, n_j)} \right). \quad (5.17)$$

It is the same as (2.26) except that now we have conditioned it upon the knowledge of the symbols being jammed $\mathbf{b}$ and hence the number of jammed symbols n_j , out of the received n -tuple vector $\mathbf{r}$.

Assuming that the received symbols are independent, $p(\mathbf{r} | \mathbf{x} = c^i, n_j)$ can be expressed as

$$p(\mathbf{r} | \mathbf{x} = c^i, n_j) = \prod_{j=1}^n p(r_j | x_j = c_j^i, b_j) \quad (5.18)$$

which can be shown to be

$$p(\mathbf{r} | \mathbf{x} = c^i, n_j) = \left(\frac{1}{\sqrt{2\pi(\sigma_n^2 + \sigma_j^2/\rho)}} \right)^{n_j} \cdot \left(\frac{1}{\sqrt{2\pi\sigma_n^2}} \right)^{n-n_j} \cdot \exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^i)^2}{2(\sigma_n^2 + \sigma_j^2/\rho)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^i)^2}{2\sigma_n^2} \right] \quad (5.19)$$

If the received symbols within a codeword are *not* independent of one another, it implies that there is correlation between the fade coefficients of the received symbols. Hence

(5.18) would not be a simple pure product of the individual pdf of the individual symbol.

Decoding performance is expected to be worse due to inter-symbol correlation.

We have separated the n symbols of a codeword into n_J jammed and $(n - n_J)$ unjammed symbols with noise PSD as given in (5.2).

Using (5.19) in (5.17),

$$\Lambda(d_j) = \ln \left(\frac{\sum_{c^i \in \{s_j^{+1}\}} \exp \left[-\sum_{l=1}^{n_J} \frac{(r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_J^2 / \rho \right)} - \sum_{l=n_J+1}^n \frac{(r_l - c_l^i)^2}{2\sigma_n^2} \right]}{\sum_{c^i \in \{s_j^{-1}\}} \exp \left[-\sum_{l=1}^{n_J} \frac{(r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_J^2 / \rho \right)} - \sum_{l=n_J+1}^n \frac{(r_l - c_l^i)^2}{2\sigma_n^2} \right]} \right). \quad (5.20)$$

Following Section 2.5.2.1, the two codewords $c^{+l(j)}$ and $c^{-l(j)}$ are also considered here to simplify the computation. Hence (5.20) can be shown to be (See Appendix D for the derivation)

$$\Lambda(d_j) = \sum_{l=1}^{n_J} \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^{+l(i)})^2}{2 \left(\sigma_n^2 + \sigma_J^2 / \rho \right)} + \sum_{l=n_J+1}^n \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^{+l(i)})^2}{2\sigma_n^2} + \ln \left(\frac{\sum_i P_i}{\sum_i Q_i} \right) \quad (5.21)$$

where

$$P_i = \exp \left(\sum_{l=1}^{n_J} \frac{(r_l - c_l^{+l(i)})^2 - (r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_J^2 / \rho \right)} + \sum_{l=n_J+1}^n \frac{(r_l - c_l^{+l(i)})^2 - (r_l - c_l^i)^2}{2\sigma_n^2} \right), \text{ with } c^i \in \{s_j^{+1}\} \quad (5.22)$$

and

$$Q_i = \exp \left(\sum_{l=1}^{n_j} \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} + \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^i)^2}{2 \sigma_n^2} \right), \text{ with } c^i \in \{s_j^{-1}\}. \quad (5.23)$$

At high SJR and high SNR, $\sigma_j \rightarrow 0$ and $\sigma_n \rightarrow 0$, $\sum_i P_i \approx \sum_i Q_i \Rightarrow \sum_i P_i / \sum_i Q_i \rightarrow 1$, thus

the third term in (5.21) tends to 0.

An approximation of $\Lambda(d_j)$ can then be obtained as

$$\Lambda(d_j) \approx \Lambda'(d_j) = \sum_{l=1}^{n_j} \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^{+l(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} + \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^{+l(i)})^2}{2 \sigma_n^2}. \quad (5.24)$$

Define

$$\sigma_l^2 = \begin{cases} \sigma_n^2, & \text{if unjammed} \\ \sigma_n^2 + \sigma_j^2 / \rho, & \text{if jammed} \end{cases}. \quad (5.25)$$

Therefore,

$$\Lambda'(d_j) = \sum_{l=1}^n \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^{+l(i)})^2}{2 \sigma_l^2} \quad (5.26)$$

which can be further simplified to

$$\Lambda'(d_j) = \frac{2}{\sigma_j^2} r_j + \sum_{l=1, l \neq j}^n \frac{2 r_l c_l^{+l(j)} p_l}{\sigma_l^2} \quad (5.27)$$

where

$$p_l = \begin{cases} 0, & \text{if } c_l^{+1(j)} = c_l^{-1(j)} \\ 1, & \text{if } c_l^{+1(j)} \neq c_l^{-1(j)} \end{cases}.$$

It can be seen that (5.27) is consistent with the SISO decoding model in [31] where the 1st term is the channel LLR values (as obtained earlier) and the 2nd term is the extrinsic information gleaned from the decoding process.

We can see that in a jammed AWGN channel, we cannot normalize $\Lambda'(d_j)$ by the noise variance since it is no longer a constant over a codeword. However, we can still normalize (5.27) by the constant 2. As such, we obtain

$$r'(d_j) = \frac{\Lambda'(d_j)}{2} = \frac{1}{\sigma_j^2} r_j + \sum_{l=1, l \neq j}^n \frac{r_l c_l^{+1(j)} p_l}{\sigma_l^2} \quad (5.28)$$

which is equivalent to

$$r'(d_j) = \sum_{l=1}^n \frac{(r_l - c_l^{-1(i)})^2 - (r_l - c_l^{+1(i)})^2}{4\sigma_l^2}. \quad (5.29)$$

To be consistent with the notations used in (2.37), we express the soft output $r'(d_j)$ from the decoder as

$$r'(d_j) = \left(\sum_{l=1}^n \frac{(r_l - c_l)^2 - (r_l - d_l)^2}{4\sigma_l^2} \right) d_j \quad (5.30)$$

where $\mathbf{c} = [c_1, c_2, \dots, c_l, \dots, c_n]$ and $\mathbf{d} = [d_1, d_2, \dots, d_l, \dots, d_n]$ have the same definitions as given in Section 2.5.2.2.

To be consistent with the form given in (5.28), for cases where there is no competing codeword, we propose to use the reliability factor β such that it is normalized by the symbol's noise variance σ_l^2 . Thus,

$$r'_j = \frac{\beta d_j}{\sigma_i^2} \quad (5.31)$$

where β can be obtained from (2.42).

The extrinsic information can therefore be expressed as

$$w_j = r'_j - \frac{r_j}{\sigma_j^2} = \begin{cases} \left(\sum_{i=1}^n \frac{(r_i - c_i)^2 - (r_i - d_i)^2}{4\sigma_i^2} \right) d_j - \frac{r_j}{\sigma_j^2} & \text{if } \mathbf{c} \text{ exists} \\ \frac{\beta}{\sigma_j^2} d_j - \frac{r_j}{\sigma_j^2} & \text{if } \mathbf{c} \text{ does not exist} \end{cases} \quad (5.32)$$

5.3.1.2 Case 2: Decoding without JSI

Since there is no knowledge of JSI, the channel reliability values can be found from (2.19) and the soft output of the decoder is given by (2.37).

5.2.3 Decoding in Pulsed Jamming Rayleigh Fading Channel

5.3.1.3 Case 1: Decoding with JSI and CSI

Channel Reliability Values:

With a_j being the fading coefficient of each received symbol r_j , the reliability value of each received symbol r_j from the Pulsed Jamming Rayleigh fading channel is given by

$$\Lambda(r_j) = \begin{cases} \frac{2}{\sigma_n^2} a_j r_j, & \text{if } b_j = 0 \\ \frac{2}{\sigma_n^2 + \sigma_j^2 / \rho} a_j r_j, & \text{if } b_j = 1 \end{cases} \quad (5.33)$$

Soft Output from the channel Decoder:

Similarly, under the assumption that SJR and SNR are large, the soft output from the decoder $r'(d_j)$ can be shown to be

$$r'(d_j) = \sum_{l=1}^n \frac{\left(r_l - a_l c_l^{-1(i)}\right)^2 - \left(r_l - a_l c_l^{+1(i)}\right)^2}{4\sigma_l^2}. \quad (5.34)$$

Using the notations used in (2.37), the soft output $r'(d_j)$ from the decoder is given by

$$r'(d_j) = \left(\sum_{l=1}^n \frac{(r_l - a_l c_l)^2 - (r_l - a_l d_l)^2}{4\sigma_l^2} \right) d_j \quad (5.35)$$

where c and d have the same definitions as given in Section 2.5.2.2.

It is also noteworthy that the same soft output will result if the following is performed

$$r'(d_j) = \left(\sum_{l=1}^n \frac{(a_l r_l - c_l)^2 - (a_l r_l - d_l)^2}{4\sigma_l^2} \right) d_j \quad (5.36)$$

Similarly, for cases where there is no competing codeword, we propose to use the normalised reliability factor such that

$$r'_j = \frac{\beta a_j d_j}{\sigma_l^2} \quad (5.37)$$

where β can be obtained from (2.42).

5.3.1.4 Case 2: Decoding without JSI and CSI

With no knowledge of JSI and CSI, the channel reliability values can be found from (2.19) and the soft output of the decoder by (2.37).

5.4 Results and Discussions

We simulate the BERs for the following cases:

- a) No utilization of JSI (and CSI for Rayleigh fading channel) in the decoding process.
- b) Partial utilization of JSI (and CSI) in the decoding process, i.e. JSI (and CSI) are used to compute the extrinsic information, but not used to determine the test patterns (tp) for use in the Pyndiah's algorithm (see Section 2.5.1.3 for information on tp)
- c) Full utilization of JSI (and CSI) in the decoding process to obtain the tp and to compute the extrinsic information.

Fig. 5.1 shows the asymptotic BER bound and its simulated values for $PC(7, 4)^2$ with $\rho = 0.3$, $E_b/N_o = 7\text{dB}$ and 4 iterations in a pulsed jamming AWGN Channel. The asymptotic BER bound is obtained using (5.6) using the $d_{\min}$ and $A_{d_{\min}}$ terms only and is hence a lower BER bound. The simulated BERs are obtained under cases a, b and c respectively. Under case a, (2.19) and (2.37) are used; under case b, (2.19) and (5.30) are used; under case c, (5.16) and (5.30) are used. In the simulations, the noise-variance weighted reliability factor from (5.31) and the α weighting factor from (2.41) are used.

Fig. 5.1. Asymptotic BER bound and simulated BER values for PC(7, 4)² in Pulsed Jamming AWGN Channel with $\rho = 0.3$ and $E_b/N_o = 7$ dB

Fig. 5.1 shows that the asymptotic bound is a good estimate for the simulated BER with full JSI. As E_b/N_j increases, the tightness of the bound also increases. At higher E_b/N_j (≥ 8 dB), the full or partial use of JSI is not important as their simulated BER are about the same. However, at lower E_b/N_j , the simulated BER of partial JSI is higher than that of full JSI. At BER of 2×10^{-4} , the coding loss in term of E_b/N_j (dB) is about 1 dB. This can be explained using (5.16). As E_b/N_j decreases, σ_j^2 becomes significant in (5.16). If this information is not used by the decoder (as in the partial JSI case), the least reliable bits due to jamming may not be correctly identified, the resultant test pattern will

hence be non-optimal. Fig. 5.1 also shows very poor BER results when JSI is not used (No JSI), as expected due to the large mismatch between the assumed and true noise PSD.

Fig. 5.2 shows the asymptotic BER bound and simulated BER values for $PC(7, 4)^2$ in a pulsed jamming AWGN Channel with $E_b/N_o = 7\text{dB}$ and a smaller $\rho = 0.05$. The simulated BERs shown are for decoding with full, partial and no JSI. The bound is again a tight estimate for the simulated BER with full JSI. Furthermore, the simulated BER with full JSI is relatively flat i.e. there is no noticeable improvement to BER even as E_b/N_j increases. This is because when ρ is very small, the probability of a symbol being jammed is very small. Hence the BER approaches that of a AWGN channel without jamming.

Fig. 5.2. Asymptotic BER bound and simulated BER values for $PC(7, 4)^2$ in Pulsed Jamming AWGN Channel with $\rho = 0.05$ and $E_b/N_o = 7\text{dB}$

Fig. 5.3 shows the asymptotic BER bound and simulated BER values for $PC(7, 4)^2$ in a pulsed jamming Rayleigh fading channel with $\rho = 0.3$, $E_b/N_o = 7\text{dB}$ and 4 iterations. The asymptotic BER bound is obtained using (5.11) using the d_{min} and $A_{d_{min}}$ terms only. The simulated BERs are obtained under decoding with full, partial and no JSI respectively. Under no JSI, (2.19) and (2.37) are used; under partial JSI, (2.19) and (5.36) are used; under full JSI, (5.33) and (5.36) are used. In the simulations, the weighted reliability factor from (5.37) and α weighting factor from (2.41) are used.

Fig. 5.3. Asymptotic BER bound and simulated BER values for PC(7, 4)² in Pulsed Jamming Rayleigh Fading Channel with $\rho = 0.3$ and $E_b/N_o = 7\text{dB}$

The result from Fig. 5.3 show that in pulsed jamming Rayleigh fading channel, decoding with partial or full use of JSI and CSI are both able to meet the lower BER bound, but not so if JSI and CSI are not used by the decoder.

Fig. 5.4 shows the asymptotic BER bound and simulated BER values for PC(7, 4)² in a pulsed jamming Rayleigh fading channel with $E_b/N_o = 7\text{dB}$, 4 iterations and a smaller $\rho = 0.05$. Similarly, the asymptotic bound is reached by decoding with full or partial JSI and CSI. Also, when full JSI and CSI are used, the simulated BER is relatively flat when ρ is very small.

Fig. 5.4. Asymptotic BER bound and simulated BER values for $PC(7, 4)^2$ in Pulsed Jamming Rayleigh Fading Channel with $\rho = 0.05$ and $E_b/N_o = 7\text{dB}$

From the BER results obtained so far, we can see that both JSI and CSI are critical in obtaining correct test patterns and soft output in the decoding process of Pyndiah’s decoding algorithm. Also, the BER bounds (5.6) and (5.11) which we derive for pulsed jamming AWGN and pulsed jamming Rayleigh fading channel respectively are good estimates of the true BER.

Fig. 5.5 shows the BER lower bound and its simulated values for a larger product code $PC(31, 26)^2$ in a pulsed jamming AWGN channel with $\rho = 0.3$, $E_b/N_o = 7\text{dB}$ and 4 iterations. Decoding with full JSI is used to obtain the simulated BER. Unlike Fig. 5.1 and 5.2, the original weighting factor α from (2.41) does not result in good BER in this

case. Hence, another set of manually optimized weighting factor given by $\alpha_{opt} = [0.0 \ 0.2 \ 0.3 \ 0.5 \ 0.7 \ 0.5 \ 0.5 \ 0.5]$ is used. The procedure for optimizing the scaling factor is as follows: As a guideline provided by [1], alpha takes a small value in the first few decoding steps and increases as BER is better. This is to reduce the effect of the extrinsic information in the initial decoding steps when the BER is relatively high. Similarly, in our manual optimization of the alpha vector set, alpha is given to be a small value in the first decoding step. For the next decoding steps (iterations), alpha is increased by a small value and the BER generated by this alpha value is observed. Alpha is then increased gradually with the corresponding BER observed. This continues until there is an increase in BER (or drop in performance). The alpha chosen for this particular decoding step is the one which generates the best BER. The same procedure is then repeated for subsequent decoding steps. The result shows that with some optimization of the weighting factor, the simulated BER is able to approach the asymptotic BER bound as E_b/N_f increases.

Fig. 5.5. Asymptotic BER bound and simulated BER values for PC(31, 26)² in Pulsed Jamming AWGN Channel with $\rho = 0.3$ and $E_b/N_o = 7\text{dB}$

Fig. 5.6 shows the asymptotic BER bound and simulated values (with full JSI and CSI) for PC(31, 26)² in a pulsed jamming Rayleigh channel with $\rho = 0.5$, $E_b/N_o = 10\text{dB}$ and 4 iterations. For both simulations (simulations which use the set of (1) manually optimised alpha values and (2) alpha values from (2.41) respectively), the normalised reliability factor from (5.37) is used. When the α weighting factor from (2.41) is used in the decoding, very poor BER performance is again obtained. Therefore, a set of manually optimized weighting factor given by $\alpha_{opt} = [0.0 \ 0.2 \ 0.7 \ 0.7 \ 0.5 \ 0.5 \ 0.5 \ 0.5]$ is used, and the resultant BER approaches the asymptotic BER bound as E_b/N_j increases.

Fig. 5.6. Asymptotic BER bound and its simulated values for PC(31, 26)² in Pulsed Jamming Rayleigh Fading Channel with $\rho = 0.5, E_b/N_o = 7\text{dB}$

The above results show that with JSI and CSI applied to obtain the test positions and to compute the extrinsic information, decoding product codes using Pyndiah’s algorithm with optimized α values is able to approach the asymptotic BER bounds.

Chapter 6

Conclusions and Recommendations

6.1 Conclusions

In this thesis, we have derived new weight enumerator expressions for conventional shortened and shortened-extended block/product codes, and shown that they are tight for Hamming, BCH and Golay product codes. The weight enumerator obtained is compact and easy to use as it is expressed in terms of known parameters of the original (unshortened) code. For small codes, the tightness of the proposed weight enumerator is verified using true weight enumerator values while for larger codes, the tightness is verified by deriving the associated BER bounds and comparing with Monte Carlo simulations.

To obtain better shortened codes than the conventional code shortening scheme (which nullifies a block of contiguous information symbol positions), we propose to shorten a code by nullifying carefully selected information symbol positions which correspond to codewords with the smallest Hamming distance, and we have devised an efficient algorithm to identify such information symbol positions using the parity check matrix. Called *ordered shortening*, the proposed scheme produces shortened codes with larger d_{min} (minimum Hamming distance) and/or smaller A_d (code weight enumerator)

values than a conventional-shortened code with the same code length and code rate. These improved d_{min} and $A_{d_{min}}$ properties lead to improved BER performance for the *ordered-shortened* codes, as verified by computer simulations using the Pyndiah decoding algorithm. Furthermore, the d_{min} values of the proposed *ordered shortening* scheme are shown to meet the theoretical upper bound values for some class of codes.

We have also extended the Pyndiah's turbo decoding algorithm for product code to pulsed jamming channels with AWGN and Rayleigh fading. The BER bound for pulsed jamming channel with Rayleigh fading is also derived. The results show that with the use of jamming state information and channel state information to obtain the test positions and to compute the extrinsic information, decoding small product codes using Pyndiah's algorithm is able to approach the asymptotic BER well. For large product codes, we found that the α weighting factor in Pyndiah's algorithm must be properly optimized in order to produce the best BER results.

6.2 Recommendations

Some suggestions for future work are given below:

- a) As shown in Section 5.4, the values of the α weighting factors used for Pyndiah's algorithm for decoding larger product codes such as $PC(31,26)^2$ in pulsed jamming channels need to be re-optimized for a larger product code as the original values given in [1] cannot give satisfactory results. More investigations can be performed to derive the general expression of the optimized α , or systematic ways to perform the optimization.

-
- b) So far we have investigated Pyndiah's decoding of product codes (5.24) in pulsed jamming channels with high E_b/N_o and E_b/N_J values. More work should be done to extend Pyndiah's decoding algorithm to the low E_b/N_J regions, which are also of interest, especially to military communications.
 - c) A shortening scheme (*ordered shortening*) for shortened code has been described in Chapter 4. Work can be extended to punctured code where a better puncturing scheme (as compared to the conventional puncturing scheme) can be designed.
 - d) The weight enumerator expressions derived in Chapter 3 are for conventional shortened and shortened-extended block/product codes. Work can be done to derive the weight enumerator for the *ordered-shortened* codes proposed in Chapter 4.
-

List of Publications

- [1] K.C. Lim and Yong Liang, Guan, “*On the Weight Enumerator and BER Bounds of Shortened Block and Product Codes*”, Paper accepted for International Conference on Information, Communications and Signal Processing 2005, Bangkok, Thailand, December 2005.
 - [2] K.C. Lim and Yong Liang, Guan, “*Ordered Shortening*”, Paper accepted for Vehicular Technology Conference 2006, Australia, Melbourne, 7-10 May 2006.
 - [3] K.C. Lim and Yong Liang, Guan, “*Ordered Shortening*”, Submitted to IEEE Comms Lett.
 - [4] K.C. Lim and Yong Liang, Guan, “*On the Weight Enumerator and BER Bounds of Shortened Block and Product Codes*”, Submitted to IEEE Comms Lett.
-

Bibliography

- [1] Pyndiah, R.M., "Near-optimum decoding of product codes: block turbo codes", *IEEE Trans. Communications*, vol. 46, pp. 1003 – 1010, Aug. 1998.
 - [2] Nam Yul Yu, Young Kim, Pil Joong Lee, "Iterative decoding of product codes composed of extended Hamming codes", *Fifth IEEE Symposium on Computers and Communications*, pp. 732 – 737, 3-6 July 2000.
 - [3] Ferrari, M., Bellini, S., "Importance Sampling Simulation of Turbo Product Codes", *IEEE International Conference on Communications*, vol. 9, pp. 2773 – 2777, 11-14 June 2001.
 - [4] P.A. Martin and D.P. Taylor, "On Multilevel Codes and Iterative Multistage Decoding", *IEEE Trans. Commun.*, Vol. 49, pp. 1916-1925, Nov 2001.
 - [5] P.A. Martin and D.P. Taylor, "Distance Based Adaptive Scaling in Sub-Optimal Iterative Decoding", *IEEE Trans. Commun.*, Vol. 50, pp. 869-871, June 2002.
 - [6] M.P.C. Fossorier and S. Lin, "Soft-Input Soft-Output Decoding of Linear Block Codes Based on Ordered Statistics," in *Proc. Globecom*, Vol. 5, Nov. 1998, pp. 2828-2823.
 - [7] P.A. Martin, D.P. Taylor and M.P.C. Fossorier, "Soft-Input Soft-Output List Based Decoding Algorithm", *IEEE Trans. Commun.*, Vol. 52, pp. 252-262, Feb 2004.
 - [8] P.A. Martin, A. Valembois, M.P.C. Fossorier and D.P. Taylor, "On Soft-Input Soft-Output Decoding using "Box and Match" Techniques", *IEEE Trans. Commun.*, Vol. 52, pp. 2033-2037, Dec. 2004.
-

-
- [9] IEEE Standard 802.16a-2003, "IEEE Standard for Local and Metropolitan Area Networks — Part 16: Air Interface for Fixed Broadband Wireless Access Systems — Amendment: Medium Access Control Modifications and Additional Physical Layer Specifications for 2–11GHz", 2003.
- [10] Proakis J. G., *Digital Communications*, 3rd ed, McGraw-Hill, 1995.
- [11] Tolhuizen, L.M.G.M., "More results on the weight enumerator of product codes", *IEEE Trans. Information Theory*, vol. 48, pp. 2573 – 2577, Sept. 2002.
- [12] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications*, Englewood Cliffs, NJ: Prentice-Hall, 1983.
- [13] Kasami, T., Fujiwara, T., Shu Lin, "An approximation to the weight distribution of binary linear codes", *IEEE Trans. Information Theory*, vol.31, pp. 769 – 780, Nov 1985.
- [14] Y. Desaki, T. Fujiwara, T. Kasami, "A method for computing the weight distribution of a block code by using its trellis diagram", *IEICE Trans. Fundamentals*, vol. E77-A, pp. 1230–1237, Aug. 1994.
- [15] Tolhuizen, L.; Baggen, S.; Hekstra-Nowacka, E., "Union bounds on the performance of product codes", *IEEE Int. Symposium, Information Theory*, pp. 267, 1998.
- [16] Cheng, U., Huth, G.K., "Bounds on the bit error probability of a linear cyclic code over $GF(2^l)$ and its extended code", *IEEE Transactions on Information Theory*, vol. 34, Issue 4, pp. 776 – 785, July 1988.
-

-
- [17] Fujiwara T., Kasami T., Kitai A., Shu Lin, "On the Undetected Error Probability for Shortened Hamming Codes", *IEEE Trans. Communications*, vol. 33, pp. 570 – 574, Jun 1985.
 - [18] A. Mahran, M. Benaissa, "Adaptive reliability calculation for block turbo codes", *Vehicular Technology Conference*, vol. 1, pp. 674 – 677, 6-9 Oct. 2003.
 - [19] C. Argon, S.W. McLaughlin, "An efficient Chase decoder for turbo product codes", *IEEE Transactions on Communications*, vol. 52, Issue 6, pp. 896-898, June 2004.
 - [20] S.A. Hirst, B. Honary, G. Markarian, "Fast Chase algorithm with an application in turbo decoding", *IEEE Transactions on Communications*, vol. 49, Issue 10, pp. 1693 – 1699, Oct. 2001.
 - [21] J.H. Kang, W.E. Stark, "Turbo codes for coherent FH-SS with partial band interference", *MILCOM 97 Proceedings*, vol. 1, pp. 5 – 9, 2-5 Nov. 1997.
 - [22] Eng Seng Chia, R.C. Robertson, "Performance of serially concatenated convolutional codes with BPSK in the presence of pulsed noise jamming", *9th Asia-Pacific Conference on Communications*, vol. 1, pp. 378 – 382, 21-24 Sept. 2003.
 - [23] Branka Vucetic, Jinhong Yuan, *Turbo codes: Principles and Applications*, Boston: Kluwer Academic Publishers, pp. 23-28, 2000.
 - [24] Kim, M. ., Lee Y. H., "Undetected error probabilities of binary primitive BCH codes for both error correction and detection", *IEEE Trans. Commun.*, vol. 44, pp.575-580, May 1996.
 - [25] Desset C., Labeau F., Vandendorpe L., Macq B., "Improved and unified approximation for the BER of linear block codes", in *Proc. EUSIPCO*, vol. 4, Tampere, Finland, Sept. 2000.
-

-
- [26] Hall, E.K., Wilson, S.G, "Design and analysis of turbo codes on Rayleigh fading channels", *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 160 – 174, Feb. 1998.
 - [27] Desset, C., Macq, B., Vandendorpe, L., "Computing the word-, symbol-, and bit-error rates for block error-correcting codes", *IEEE Trans Commun*, vol. 52, pp. 910 - 921, June 2004.
 - [28] Hermann J. Helgert, Russell D. Stinaff, "Minimum-Distance Bounds for Binary Linear Codes", *IEEE Transactions on Information Theory*, vol. 19, Issue 3, pp. 344 – 356, May 1973.
 - [29] Tom Verhoeff, "An Updated Table of Minimum-Distance Bounds for Binary Linear Codes", *IEEE Transactions on Information Theory*, vol. 33, Issue: 5, pp. 665 – 680, Sep 1987.
 - [30] A.E. Brouwer, Tom Verhoeff, "An Updated Table of Minimum-Distance Bounds for Binary Linear Codes", *IEEE Transactions on Information Theory*, vol. 39, Issue: 2, pp. 662 – 677, March 1993.
 - [31] B.R. Vojcic, R.L. Pickholtz, "Performance of coded direct sequence spread spectrum in a fading dispersive channel with pulsed jamming", *IEEE Journal on Selected Areas in Communications*, vol. 8, Issue 5, pp 934 – 942, June 1990.
 - [32] J. Hagenauer, E. Offer, L.Papke, "Iterative decoding of binary block and convolutional codes", *IEEE Transactions on Information Theory*, vol. 42, Issue 2, pp. 429 – 445, March 1996.
 - [33] A.P. Prudnikov, Yu. A. Brychkov, O.I. Marichev, *Integrals and Series, Vol. 1: Elementary Functions*, New York: Gordon and Breach, 1986.
-

-
- [34] Baum, C.W., Raghavan, A.R., "Suboptimal Viterbi decoding metrics for channels with pulse jamming", *IEEE Proceedings, MILCOM*, vol. 3, pp. 1032 – 1035, 21-24 Oct. 1996.
- [35] P.A. Martin and D.P. Taylor, "On Adaptive Reduced-Complexity Iterative Decoding", in *Proc. Globecom*, 2000, pp. 772-776.
-

Appendix A - Verification of Derived Weight Spectrum for Shortened Hamming(31, 26)

Table A. Comparison of True and Approximated $A_{d(s)}$ for Hamming(31, 26)

Hamming(31, 26)		d													
		0	3	4	5	6	7	8	9	10	11	12	13	14	
1-2		Approximated $A_{d(s)}$ Agree Exactly With True $A_{d(s)}$													
3	True $A_{d(s)}$	1	113	706	3012	11548	37202	97647	215380	409244	671823	951708	1169160	1252728	
	App. $A_{d(s)}$	1	113	706	3012	11548	37200	97649	215388	409236	671806	951725	1169187	1252701	
4	True $A_{d(s)}$	1	101	605	2473	9075	27907	69740	146135	263109	407922	543786	626298	626430	
	App. $A_{d(s)}$	1	101	605	2474	9073	27900	69749	146156	263081	407882	543843	626350	626350	
5	True $A_{d(s)}$	1	90	515	2013	7062	20680	49060	97405	165704	241756	302030	324730	301700	
	App. $A_{d(s)}$	1	90	516	2016	7057	20667	49083	97437	165643	241708	302135	324774	301576	
6	True $A_{d(s)}$	1	79	436	1632	5430	15085	33975	63760	101944	139350	162680	162512	139188	
	App. $A_{d(s)}$	1	79	436	1629	5428	15102	33981	63709	101934	139447	162688	162387	139189	
7	True $A_{d(s)}$	1	69	366	1311	4129	10836	23094	40876	61188	77910	84560	78162	61278	
	App. $A_{d(s)}$	1	70	366	1303	4126	10874	23107	40774	61161	78090	84598	77946	61243	
8	True $A_{d(s)}$	1	60	305	1042	3096	7656	15402	25600	35672	42112	42322	35924	25480	
	App. $A_{d(s)}$	1	61	305	1031	3094	7702	15405	25484	35677	42299	42299	35725	25518	
9	True $A_{d(s)}$	1	52	252	817	2288	5320	10046	15596	20160	21952	20244	15638	9968	
	App. $A_{d(s)}$	1	53	252	807	2287	5358	10046	15512	20165	22069	20230	15533	9985	
10	True $A_{d(s)}$	1	45	205	632	1672	3620	6370	9240	11032	10934	9170	6440	3640	
	App. $A_{d(s)}$	1	46	206	624	1663	3653	6393	9166	10999	11034	9195	6354	3631	

s

Hamming(31, 26)		d													
		15	16	17	18	19	20	21	22	23	24	25	26	27	
s	1-2	Approximated $A_{d(s)}$ Agree Exactly With True $A_{d(s)}$													
	3	True $A_{d(s)}$	1170876	951279	670536	409816	216095	97218	36916	11756	3090	641	100	12	1
		App. $A_{d(s)}$	1170844	951311	670563	409789	216078	97235	36924	11748	3088	643	100	12	1
	4	True $A_{d(s)}$	543654	407625	263406	146410	69465	27753	9229	2527	551	90	11	1	0
		App. $A_{d(s)}$	543606	407705	263436	146353	69454	27781	9231	2518	551	92	11	1	0
	5	True $A_{d(s)}$	241624	166001	97570	48840	20570	7183	2057	470	80	10	1	0	0
		App. $A_{d(s)}$	241603	166102	97569	48784	20579	7203	2051	466	82	10	1	0	0
	6	True $A_{d(s)}$	102106	63895	33840	15000	5515	1668	400	70	9	1	0	0	0
		App. $A_{d(s)}$	102217	63885	33774	15011	5540	1662	394	72	9	1	0	0	0
	7	True $A_{d(s)}$	40708	22977	10908	4212	1293	330	71	9	0	0	0	0	0
		App. $A_{d(s)}$	40887	22999	10808	4203	1330	332	63	9	1	0	0	0	0
8	True $A_{d(s)}$	15192	7701	3216	1032	260	61	10	0	0	0	0	0	0	
	App. $A_{d(s)}$	15332	7666	3152	1051	277	55	8	1	0	0	0	0	0	
9	True $A_{d(s)}$	5272	2345	844	224	44	8	1	0	0	0	0	0	0	
	App. $A_{d(s)}$	5333	2333	822	228	48	7	1	0	0	0	0	0	0	
10	True $A_{d(s)}$	1652	637	200	40	5	1	0	0	0	0	0	0	0	
	App. $A_{d(s)}$	1697	636	187	42	7	1	0	0	0	0	0	0	0	

Appendix B – Verification of Derived Weight Spectrum for Shortened BCH(63, 18)

Table B. Comparison of True and Approximated $A_{d(s)}$ for BCH(63, 18)

BCH(63, 18)		d												
		0	21	22	23	24	25	26	27	28	29	30	31	
1	True $A_{d(s)}$	Approximated $A_{d(s)}$ Agree Exactly With True $A_{d(s)}$												
	App. $A_{d(s)}$													
2	True $A_{d(s)}$	1	640	1164	755	1195	3538	4898	1770	2150	9410	10038	3856	
	App. $A_{d(s)}$	1	640	1164	752	1198	3544	4892	1776	2144	9392	10056	3856	
3	True $A_{d(s)}$	1	416	748	468	730	2104	2788	992	1152	4912	5144	1896	
	App. $A_{d(s)}$	1	420	744	470	725	2088	2811	987	1163	4937	5101	1896	
4	True $A_{d(s)}$	1	267	477	290	436	1241	1567	544	616	2518	2594	916	
	App. $A_{d(s)}$	1	273	471	290	435	1218	1593	543	620	2551	2551	917	
5	True $A_{d(s)}$	1	171	301	173	259	728	862	296	328	1262	1298	442	
	App. $A_{d(s)}$	1	176	296	177	258	702	891	294	326	1297	1254	435	
6	True $A_{d(s)}$	1	109	189	100	151	424	470	165	171	609	631	209	
	App. $A_{d(s)}$	1	112	183	107	151	399	491	157	169	648	605	202	
7	True $A_{d(s)}$	1	66	119	58	88	246	245	94	83	290	311	90	
	App. $A_{d(s)}$	1	71	113	64	88	224	267	83	86	319	287	92	
8	True $A_{d(s)}$	1	38	74	32	49	144	129	54	41	132	145	38	
	App. $A_{d(s)}$	1	44	68	38	50	124	143	43	43	154	133	41	
9	True $A_{d(s)}$	1	23	46	17	27	81	63	30	18	58	72	17	
	App. $A_{d(s)}$	1	27	41	22	28	68	76	22	21	73	61	18	
10	True $A_{d(s)}$	1	13	28	10	14	47	30	16	8	23	36	6	
	App. $A_{d(s)}$	1	17	24	13	16	36	39	11	10	34	27	8	

s

BCH(63, 18)		d																
		32	33	34	35	36	37	38	39	40	41	42	63					
1	True $A_{d(s)}$ App. $A_{d(s)}$	Approximated $A_{d(s)}$ Agree Exactly With True $A_{d(s)}$																
2	True $A_{d(s)}$ App. $A_{d(s)}$	3615	8270	6810	1366	986	2390	1510	445	245	328	156	0					
		3615	8288	6792	1360	992	2384	1516	448	242	328	156	0					
3	True $A_{d(s)}$ App. $A_{d(s)}$	1719	3808	2984	576	416	944	572	164	78	104	52	0					
		1719	3796	3014	582	404	940	569	161	84	107	49	0					
4	True $A_{d(s)}$ App. $A_{d(s)}$	803	1722	1278	240	168	367	209	58	24	29	19	0					
		802	1708	1306	243	162	361	209	56	28	34	15	0					
5	True $A_{d(s)}$ App. $A_{d(s)}$	361	760	532	96	64	143	73	17	11	8	6	0					
		367	753	553	99	63	134	74	19	9	10	4	0					
6	True $A_{d(s)}$ App. $A_{d(s)}$	160	339	221	31	25	54	24	7	4	1	1	0					
		165	324	229	39	24	49	26	6	3	3	1	0					
7	True $A_{d(s)}$ App. $A_{d(s)}$	74	146	87	10	9	20	6	4	1	0	0	0					
		72	137	92	15	9	17	9	2	1	1	0	0					
8	True $A_{d(s)}$ App. $A_{d(s)}$	34	64	35	2	3	6	1	2	0	0	0	0					
		31	56	36	6	3	6	3	1	0	0	0	0					
9	True $A_{d(s)}$ App. $A_{d(s)}$	18	29	11	1	0	0	0	0	0	0	0	0					
		13	22	14	2	1	2	1	0	0	0	0	0					
10	True $A_{d(s)}$ App. $A_{d(s)}$	9	13	2	0	0	0	0	0		0	0	0					
		5	9	5	1	0	1	0	0	0	0	0	0					

s

Appendix C – Derivation of (5.10)

Substituting (5.8) and (2.13) into (5.9),

$$\begin{aligned}\bar{P}_d(d, r_c | n_J) &= \int \cdots \int_{a_1} P_d(d, r_c | n_J, (a_1, a_2, \dots, a_d)) \cdot p_a(a_1, a_2, \dots, a_d) da_1 \cdots da_d \\ &= \int \cdots \int_{a_1} \int_{a_d} \frac{1}{\pi} \int_0^{\pi/2} e^{-\frac{\left(\left(\frac{r_c E_b}{N_J / \rho + N_0} \right) \sum_{i=1}^{n_J} a_i^2 + \frac{r_c E_b}{N_0} \sum_{i=n_J+1}^d a_i^2 \right)}{\sin^2 \phi}} d\phi \cdot \prod_{i=1}^d p(a_i) da_1 \cdots da_d\end{aligned}$$

Since the fading coefficients a_i are i.i.d, the above expression can be reduced to a product of integral over each a_i ,

$$\begin{aligned}\bar{P}_d(d, r_c | n_J) &= \frac{1}{\pi} \int_0^{\pi/2} \left[\int \cdots \int_{a_1} \prod_{i=1}^d p(a_i) \cdot e^{-\frac{\left(\left(\frac{r_c E_b}{N_J / \rho + N_0} \right) \sum_{i=1}^{n_J} a_i^2 + \frac{r_c E_b}{N_0} \sum_{i=n_J+1}^d a_i^2 \right)}{\sin^2 \phi}} da_1 \cdots da_d \right] d\phi \\ &= \frac{1}{\pi} \int_0^{\pi/2} \left[\int_{a_i} p(a_i) \cdot e^{-\frac{\left(\left(\frac{r_c E_b}{N_J / \rho + N_0} \right) \sum_{i=1}^{n_J} a_i^2 + \frac{r_c E_b}{N_0} \sum_{i=n_J+1}^d a_i^2 \right)}{\sin^2 \phi}} da_i \right]^d d\phi\end{aligned}$$

Let

$$U = \int_{a_i} p(a_i) \cdot e^{-\frac{\left(\left(\frac{r_c E_b}{N_J / \rho + N_0} \right) \sum_{i=1}^{n_J} a_i^2 + \frac{r_c E_b}{N_0} \sum_{i=n_J+1}^d a_i^2 \right)}{\sin^2 \phi}} da_i$$

Each received symbol can be classified into one of 2 cases: jammed or unjammed.

Firstly, consider an unjammed symbol. U becomes

$$U = \int_{a_i} p(a_i) \cdot e^{-\frac{\frac{r_c E_b}{N_0} a_i^2}{\sin^2 \phi}} da_i$$

with $p(a_i)$ from (2.14) which evaluates to U_1 :

$$U_1 = \frac{\sin^2 \phi}{\frac{r_c E_b}{N_0} + \sin^2 \phi}$$

Now consider a jammed symbol. U becomes

$$U = \int_{a_i} p(a_i) \cdot e^{-\frac{\left(\frac{r_c E_b}{N_J / \rho + N_0}\right) a_i^2}{\sin^2 \phi}} da_i$$

which evaluates to U_2 :

$$U_2 = \frac{\sin^2 \phi}{\frac{r_c E_b}{N_J / \rho + N_0} + \sin^2 \phi}$$

Since there are n_J jammed symbols and $(n - n_J)$ unjammed symbols,

$$\begin{aligned} \bar{P}_d(d, r_c | n_J) &= \frac{1}{\pi} \int_0^{\pi/2} \left[\int_{a_i} p(a_i) \cdot e^{-\frac{\left(\left(\frac{r_c E_b}{N_J / \rho + N_0}\right) \sum_{i=1}^{n_J} a_i^2 + \frac{r_c E_b}{N_0} \sum_{i=n_J+1}^d a_i^2\right)}{\sin^2 \phi}} da_i \right]^d d\phi \\ &= \frac{1}{\pi} \int_0^{\pi/2} U_1^{d-n_J} U_2^{n_J} d\phi \\ &= \frac{1}{\pi} \int_0^{\pi/2} \left(\frac{\sin^2 \phi}{\frac{r_c E_b}{N_J / \rho + N_0} + \sin^2 \phi} \right)^{n_J} \left(\frac{\sin^2 \phi}{\frac{r_c E_b}{N_0} + \sin^2 \phi} \right)^{d-n_J} d\phi \end{aligned}$$

Appendix D – Derivation of (5.21)

$$\begin{aligned}
\Lambda(d_j) &= \ln \left[\frac{\sum_{c^i \in \{s_j^{-1}\}} \exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^i)^2}{2 \sigma_n^2} \right]}{\sum_{c^i \in \{s_j^{-1}\}} \exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^i)^2}{2 \sigma_n^2} \right]} \right. \\
&\quad \cdot \left. \frac{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{-1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-1(i)})^2}{2 \sigma_n^2} \right] \exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{+1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{+1(i)})^2}{2 \sigma_n^2} \right]}{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{+1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{+1(i)})^2}{2 \sigma_n^2} \right] \exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{-1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-1(i)})^2}{2 \sigma_n^2} \right]} \right] \\
&= \ln \left[\frac{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{+1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{+1(i)})^2}{2 \sigma_n^2} \right]}{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{-1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-1(i)})^2}{2 \sigma_n^2} \right]} \right] + \ln \left(\frac{\sum_i P_i}{\sum_i Q_i} \right) \\
&= \sum_{l=1}^{n_j} \frac{(r_l - c_l^{-1(i)})^2 - (r_l - c_l^{+1(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} + \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-1(i)})^2 - (r_l - c_l^{+1(i)})^2}{2 \sigma_n^2} + \ln \left(\frac{\sum_i P_i}{\sum_i Q_i} \right)
\end{aligned}$$

where

$$\begin{aligned}
P_i &= \frac{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^i)^2}{2 \sigma_n^2} \right]}{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{+l(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{+l(i)})^2}{2 \sigma_n^2} \right]} \text{ with } c^i \in \{s_j^{+1}\} \\
&= \exp \left(\sum_{l=1}^{n_j} \frac{(r_l - c_l^{+l(i)})^2 - (r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} + \sum_{l=n_j+1}^n \frac{(r_l - c_l^{+l(i)})^2 - (r_l - c_l^i)^2}{2 \sigma_n^2} \right)
\end{aligned}$$

and

$$\begin{aligned}
Q_i &= \frac{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^i)^2}{2 \sigma_n^2} \right]}{\exp \left[-\sum_{l=1}^{n_j} \frac{(r_l - c_l^{-l(i)})^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} - \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-l(i)})^2}{2 \sigma_n^2} \right]} \text{ with } c^i \in \{s_j^{-1}\} \\
&= \exp \left(\sum_{l=1}^{n_j} \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^i)^2}{2 \left(\sigma_n^2 + \sigma_j^2 / \rho \right)} + \sum_{l=n_j+1}^n \frac{(r_l - c_l^{-l(i)})^2 - (r_l - c_l^i)^2}{2 \sigma_n^2} \right)
\end{aligned}$$