Reliability-Based Coded Modulation for Low-Density Parity-Check Codes

by

Robert D. Maddock

A thesis submitted to the
Faculty of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Applied Science

Ottawa-Carleton Institute For Electrical and Computer Engineering
Department of Systems and Computer Engineering
Faculty of Engineering
Carleton University
Ottawa, Ontario
August, 2004

Copyright © 2004 by Robert D. Maddock

Library and
Archives Canada

Bibliothèque et
Archives Canada

Published Heritage
Branch

Direction du
Patrimoine de l'édition

395 Wellington Street
Ottawa ON K1A 0N4
Canada

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence

ISBN: 0-612-97424-3

Our file Notre référence

ISBN: 0-612-97424-3

The author has granted a non-exclusive license allowing the Library and Archives Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

PAGINATION ERROR.

ERREUR DE PAGINATION.

TEXT COMPLETE.

LE TEXTE EST COMPLET.

Abstract

Since the invention of Trellis-Coded Modulation (TCM) in 1982 by Gottfried Ungerboeck it has been known that significant gains can be achieved by jointly designing coding and modulation. This paper presents a comparison of the current modulation schemes used with Low-Density Parity-Check (LDPC) codes, and introduces a new coded modulation scheme for LDPC codes entitled *Reliability-Based Coded Modulation* (RBCM). This new method can improve Bit Error Rate (BER) performance of regular LDPC codes for non-binary modulation schemes by 0.2 to 0.6 dB, with little or no added complexity.

Acknowledgements

I would like to thank my thesis supervisor, Professor Amir Banihashemi, for his unwavering support and counsel. He has been an invaluable ally and friend. I would like to thank Carleton University, specifically the Department of Systems and Computer Engineering, for giving me the opportunity to study with such talented peers. I would like to thank the Ontario Graduate Scholarship in Science and Technology (OGS-ST) program for their much needed financial contribution, without whose help I would not have been able to produce this document. Lastly I would like to profoundly thank my family and friends who have been so supportive.

This document was created using $\text{\LaTeX 2}_{\epsilon}$, and compiled under MiKTeX . I would briefly like to express my thanks to the creators of $\text{\TeX}$ and $\text{\LaTeX 2}_{\epsilon}$, Donald Knuth and Leslie Lamport respectively, and others who have helped create and distribute $\text{\LaTeX 2}_{\epsilon}$. Their work in standardizing word processing and document formats has been outstanding.

Contents

Abstract	iii
Acknowledgements	iv
List of Tables	ix
List of Figures	xi
List of Acronyms	xii
1 Introduction	1
1.1 Research Motivation	2
1.2 Research Contributions	3
1.3 Organization of Thesis	4
2 Background	5
2.1 Low-Density Parity-Check Codes	5
2.1.1 Tanner Graphs	6
2.1.2 Iterative Decoding of LDPC Codes	8
2.1.3 Belief Propagation Algorithm	9

2.2	Multi-Level Coding	13
2.2.1	Multi-Stage Decoding	14
2.2.2	Parallel Independent Decoding	15
2.2.3	MLC and LDPC	17
2.3	Bit-Interleaved Coded Modulation	18
2.3.1	BICM and LDPC Codes	19
2.4	MLC and BICM	20
2.5	Channel Adapters	20
3	Modulation Schemes	22
3.1	4-PAM	24
3.1.1	LLR Analysis	24
3.2	8-PSK	33
3.2.1	LLR Analysis	33
3.3	16-PSK	40
3.3.1	LLR Analysis	40
3.4	64-PSK	45
3.4.1	LLR Analysis	45
4	Reliability-Based Coded Modulation	50
4.1	Variable Node Optimization	51
4.1.1	Implementation	52
4.2	Check Node Optimization	53
4.2.1	Implementation	55
5	Simulation Results	58

<i>CONTENTS</i>	vii
5.1 LDPC Codes Used	59
5.2 Regular Codes	62
5.2.1 Results for reg(1008,504)	63
5.2.2 Results for reg(1920,1280)	66
5.3 Irregular Codes	69
5.3.1 Results for PEG(504,252)	70
5.3.2 Results for PEG(1008,504)	74
6 Conclusion	79
6.1 Future Work	80
A Additional Tables	81
A.1 16-PSK Gray Mapping	81
A.2 64-PSK Gray Mapping	81

List of Tables

3.1	Mean of LLR, 4-PAM Gray Mapping	25
3.2	Probability that $\text{LLR} < 0$, for 4-PAM Gray Mapping	25
3.3	Absolute value of lower limit of $\text{LLR}'(c_1)$	27
3.4	Distance Spectrum for 4-PAM (GM)	27
3.5	Peak values of LLR, 4-PAM Gray Mapping	29
3.6	Mean of LLR, 8-PSK Gray Mapping	33
3.7	Probability that $\text{LLR} < 0$, for 8-PSK Gray Mapping	34
3.8	Distance Spectrum for 8-PSK (GM) [1]	35
3.9	Mean of LLR, 16-PSK Gray Mapping	40
3.10	Probability that $\text{LLR} < 0$, for 16-PSK Gray Mapping	41
3.11	Mean of LLR, 64-PSK Gray Mapping	45
3.12	Probability that $\text{LLR} < 0$, for 64-PSK Gray Mapping	46
5.1	Degree Distribution of $\text{reg}(1008,504)$	59
5.2	Degree Distribution of $\text{reg}(1920,1280)$	60
5.3	Degree Distribution of $\text{PEG}(504,252)$	61
5.4	Degree Distribution of $\text{PEG}(1008,504)$	62
5.5	$\text{PEG}(504,252)$ Unreliable Column Weight Distribution, 64-PSK	72

5.6	PEG(504,252) Unreliable Column Weight Distributions, 4-PAM . . .	74
5.7	PEG(1008,504) Unreliable Column Weight Distributions, 8-PSK . . .	77

List of Figures

2.1	Parity-Check Matrix of a (10,5) Block Code	7
2.2	Tanner Graph of the (10,5) Block Code	7
2.3	MLC Encoder	14
2.4	MSD Decoder	16
2.5	PID Decoder	17
2.6	BICM Encoder & Decoder	19
3.1	LDPC Decoder and Channel Adapter	23
3.2	Signal Constellation for 4-PAM Gray Mapping	24
3.3	Histogram of LLRs for 4-PAM Modulation, with Gray Mapping . . .	30
3.4	Signal Constellation for 8-PSK Gray Mapping [1]	34
3.5	Histogram of LLRs for 8-PSK Modulation, with Gray Mapping . . .	37
3.6	Histogram of LLRs for 16-PSK Modulation, with Gray Mapping . . .	42
3.7	Histogram of LLRs for 64-PSK Modulation, with Gray Mapping . . .	47
5.1	Probability of Error Curves Using reg(1008,504), 64-PSK	64
5.2	Avg. Number of Iterations vs. BER Using reg(1008,504), 64-PSK . .	65
5.3	Probability of Error Curves Using reg(1008,504), 8-PSK	67
5.4	Probability of Error Curves Using reg(1920,1280), 16-PSK	68

5.5	Probability of Bit Error Curves Using PEG(504,252), 64-PSK	71
5.6	Probability of Bit Error Curves Using PEG(504,252), 4-PAM	73
5.7	Probability of Bit Error Curves Using PEG(504,252), 8-PSK	75
5.8	Probability of Bit Error Curves Using PEG(1008,504), 64-PSK	76
5.9	Probability of Bit Error Curves Using PEG(1008,508), 8-PSK	78

List of Acronyms

LDPC:	Low-Density Parity-Check
TG:	Tanner Graph
BER:	Bit Error Rate
WER:	Word Error Rate
PSK:	Phase-Shift Keying
PAM:	Pulse Amplitude Modulation
QAM:	Quadrature Amplitude Modulation
GM:	Gray Mapping
MLC:	Multi-Level Coding
PID:	Parallel Independent Decoding
MSD:	Multi-Stage Decoding
BICM:	Bit-Interleaved Coded Modulation
LLR:	Log-Likelihood Ratio
TCM:	Trellis-Coded Modulation
i.i.d:	independent and identically distributed
RBCM:	Reliability Based Coded Modulation
VNO:	Variable Node Optimization
CNO:	Check Node Optimization

Chapter 1

Introduction

Since the invention of Trellis-Coded Modulation (TCM) in 1982 by Gottfried Ungerboeck [2, 3] it has been known that significant gains can be achieved by jointly designing coding and modulation. Coding theory has undergone major improvements since the early 80s. Low-Density Parity-Check codes has re-emerged as a powerful coding scheme and has been shown to outperform the Turbo Codes [4], and reach within 0.0045 dB of the Shannon Limit [5].

LDPC codes were originally discovered by Gallager in 1962 [6, 7], however they were largely ignored by the coding community because of their computational complexity. Their value was not appreciated until MacKay and Neil rediscovered them in 1996 [8, 9].

Higher order modulation schemes can improve spectral efficiency by transmitting more bits per symbol. In doing so, these modulation schemes can offer a different level of protection for each bit in a symbol. Some bits are further apart in the signal constellation than others leading to greater statistical reliability for these bits.

This document will introduce a new coded modulation scheme, Reliability-Based

Coded Modulation (RBCM). It improves the performance of LDPC codes by utilizing the known difference in protection levels to permute the parity-check matrix.

1.1 Research Motivation

While there are many coded modulation schemes for other codes there are few schemes developed for LDPC codes. Most existing LDPC coded modulation schemes optimize coding and modulation separately.

One of the few LDPC coded modulation schemes that jointly optimize coding and modulation is Multi-Level Coding (MLC) [10–12]. In it component code rates are adjusted to compensate for the difference in protection offered by modulation, such that each bit is equally protected overall. Another scheme used with LDPC codes is Bit Interleaved Coded Modulation (BICM) [13, 14], where coding and modulation are largely separate and divorced from one and other. However, as is shown in [12], jointly optimized MLC fails to outperform BICM. Thus BICM became the focus of research.

For regular LDPC coded BICM however, to the best of our knowledge, no design approach other than simply mixing them with different modulations through a random interleaver is proposed. In fact in [14] made the observation that a fixed correspondence between coded bits and label positions in a BICM scheme introduces unequal error protection. The authors also added that this effect is usually undesirable if the code is random. So, they concluded that for avoiding such a fixed correspondence, an ideal bit interleaver should be employed.

RBCM was designed to improve upon the performance of BICM by taking advantage of the unequal bit protection. Using this information to jointly design coding

and modulation, and thus improve performance, is the goal of this research.

1.2 Research Contributions

The major contributions of this research are the creation of a new coded modulation scheme for LDPC codes, *Reliability-Based Coded Modulation* (RBCM) and the demonstration of its ability to improve Bit Error Rate and Word Error Rate performance. In order for RBCM to be developed a detailed analysis of various modulation schemes was done to show how these schemes offered different levels of protection to each bit in a symbol. This work shows that RBCM successfully improves BER performance by capitalizing on differences in bit reliability to jointly optimize coding based on LDPC Codes and modulation using BICM.

Reliability-Based Coded Modulation is based on the idea that not all bits are equally protected by modulation, and thus some bits will be statistically more reliable than others. RBCM, developed in this thesis, permutes the parity-check matrix of the code based on the type of modulation scheme to improve the performance of the code. This method can significantly improve the BER performance of regular LDPC codes, and since it is only a one-time adjustment to the parity-check matrix it does not add *any additional complexity or delay* to the system.

We demonstrate by computer simulation that RBCM can substantially improve BER for regular LDPC codes over AWGN channel. We also show that although RBCM can be performed for irregular LDPC codes, it does not yield greater performance.

Furthermore, the results of this work also suggest that other methods could be developed based on the ideas herein, i.e., one could use the reliability information

provided by modulation schemes in other ways to improve performance.

1.3 Organization of Thesis

The thesis is organized as follows:

Chapter 2 provides background information about LDPC codes, discusses two coded modulation schemes, and explores other related concepts.

Chapter 3 provides detailed analysis of four different modulation schemes: 4-PAM, 8-PSK, 16-PSK & 64-PSK. It covers spectral efficiency, signal constellation and signal mappings. It also shows the varying levels of protection offered to each bit, for each scheme.

Chapter 4 introduces *Reliability-Based Coded Modulation* explaining the theory behind RBCM as well as implementation details.

Chapter 5 discusses the performance of computer simulations, simulation details such as codes distributions, algorithms used, maximum iterations allowed, etc. It also shows how RBCM outperforms traditional BICM for regular LDPC codes as well as how irregular codes perform.

Chapter 6 summarizes the work done and suggests some avenues for future work in this topic.

Chapter 2

Background

This chapter will discuss some necessary background, including an introduction to Low-Density Parity-Check(LDPC) coding theory and two coded modulation schemes. It is assumed that the reader has some understanding of coding and modulation theory, and in particular of LDPC codes.

2.1 Low-Density Parity-Check Codes

Low-Density Parity-Check (LDPC) codes are linear block codes whose parity-check matrix, by definition, contains mostly ‘zeros’ and very few ‘ones’. LDPC codes were discovered by Gallager in 1962 [6, 7] and introduced in his Ph.D dissertation. They were, however, almost completely ignored by the coding community for over thirty years. It was not until 1996, when MacKay and Neil re-discovered them [8, 9], that their value became apparent.

There are two broad categories of LDPC codes, *Regular* and *Irregular*. *Regular* LDPC codes are codes whose parity-check matrix contains the same number of

‘ones’ in each column, and the same number of ‘ones’ in each row¹. Specifically, the weight of each column in the m by n parity-check matrix, H , must equal W_c , and the weight of each row must equal W_r . Note that $W_r = W_c \cdot \frac{n}{m}$ where $W_c \ll m$. *Irregular* LDPC codes are codes whose column and row weights are not all equal to W_c and W_r , respectively. All of the codes originally proposed by Gallager [6, 7] were regular. Irregular codes were introduced in [15] and further studied in many papers, including [16–18].

It has been shown in [17] that irregular codes generally outperform regular codes. It has also been shown in [4] that long irregular codes can outperform the best known Turbo Codes [19], and can even achieve a performance within 0.0045 dB of the Shannon limit [5].

2.1.1 Tanner Graphs

Professor Michael Tanner was one of the very few members of the academic community who studied LDPC codes before their re-discovery [20]. He showed in [20]² that LDPC codes, and a generalization thereof, could be represented by a “bipartite graph”, now commonly called a “Tanner Graph” (TG).

LDPC Tanner Graph: A bipartite graph that contains *variable nodes* (*bit nodes*) and *check nodes*, each respectively representing the columns and rows of the parity-check matrix, H . There is an edge between the two nodes, χ_i and f_j if and only if there is a ‘one’ at position (j, i) in H .

¹This document will only deal with binary LDPC codes.

²For this paper, Tanner was awarded the 50th anniversary “Golden Jubilee Award” by the Information Society of the IEEE.

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

Figure 2.1: Parity-Check Matrix of a (10,5) Block Code

Figure 2.1 shows the parity-check matrix for a Regular (10,5) Block Code, where $W_c = 2$ and $W_r = 4$ [21]. The Tanner Graph of the same code is shown in Figure 2.2, where circles represent *variable nodes* and squares represent *check nodes*. In this particular case, since it is a binary code, the *variable nodes* are *bit nodes*. LDPC codes can be represented by either format, and thus these representations are interchangeable.

Figure 2.2: Tanner Graph of the (10,5) Block Code

The parity-check matrix, H , of an LDPC code, or equivalently the Tanner Graph of an LDPC code, defines the parity-check equations. Since LDPC codes are linear block codes, it follows that $\mathbf{c}H^T = \mathbf{0}$, where $\mathbf{c}$ is any valid codeword. Each c_i is associated to a variable node χ_i , where $i \in \{1, 2, 3, \dots, n-1\}$. In this document it

is therefore equivalent for a bit, c_i , to be associated to a check node, or a variable node, χ_i , to be associated to a check node. The bits (or variable nodes) associated to every parity-check equation, or every check node, must sum to zero. For example, the parity-check equations for (10,5) Block Code used in Figures 2.1 and 2.2 are:

$$c_0 + c_1 + c_8 + c_9 = 0 \pmod{2}$$

$$c_2 + c_5 + c_6 + c_9 = 0 \pmod{2}$$

$$c_0 + c_3 + c_5 + c_7 = 0 \pmod{2}$$

$$c_1 + c_4 + c_6 + c_7 = 0 \pmod{2}$$

$$c_2 + c_3 + c_4 + c_8 = 0 \pmod{2}$$

2.1.2 Iterative Decoding of LDPC Codes

In addition to introducing LDPC codes, Gallager also introduced an optimal³ way to decode them [6]. He proposed an iterative decoding algorithm which results in a *posteriori probability* (APP) decoding on cycle-free graphs. Since that time however, many new algorithms have come to light. These algorithms often came from many different fields but can still be used to decode LDPC codes. They are all *Message Passing Algorithms* or *Two-Way Algorithms* [22].

Message passing algorithms, when they are used to decode LDPC codes, work by exchanging information between variable nodes and check nodes iteratively, in both directions along the edges of the graph. These messages can contain either soft or hard information and help each node to determine its correct state, though the exact

³Optimal on cycle free graphs

use of the messages varies from algorithm to algorithm. Since messages are passed on each edge, the complexity of decoding is directly related to the total number of edges in the graph, which for LDPC codes is low due to the sparse nature of H .

2.1.3 Belief Propagation Algorithm

The Belief Propagation (BP) algorithm is an iterative message passing algorithm. Information flows from variable nodes to check nodes and back again, until either a valid codeword is reached or a maximum number of iterations is reached (200 to 1000 is reasonable).

The following will briefly explain the BP algorithm, using the notation used by Dr. MacKay [23, 24] and Dr. Ryan [21]. This section is derived primarily from work by Dr. Ryan [21] (used with permission), as well as from work by Dr. MacKay [23, 24].

Notations [21]:

- R_j = set of column indexes where, for the j^{th} row, $h_{ji} = 1$.
- $R_{j/i} = R_j$ excluding index i .
- C_i = set of row indexes where, for the j^{th} row, $h_{ji} = 1$.
- $C_{i/j} = C_i$ excluding index j .
- $P_i = Pr(c_i = 1|y_i)$.
- c_{kj} = the k^{th} bit in the j^{th} parity-check equation.
- y_{kj} = channel sample associated to c_{kj} .
- $P_{kj} = Pr(c_{kj} = 1|y_{kj})$.

- $S_i = \text{true}$ if $\hat{\mathbf{c}}$ satisfies all parity-check equations involving c_i .
- $q_{ij}(b)$ = message passed from variable node χ_i to check node f_j . It is the probability that $c_i = b$ given the channel sample y_i and all check node messages except f_j .
- $r_{ji}(b)$ = message passed from check node f_j to variable node χ_i . $r_{ji}(b)$ is the probability that check equation j is satisfied, given that $c_i = b$ and that the other bits are distributed according to $\{q_{ij'}\}_{j' \neq j}$.

Initially, the variable node χ_i only has information from the channel. It passes this information to all associated check nodes, $\{f_j\}_{j \in C_i}$. A check node is satisfied if the sum of all associated c_i is zero, or in other words that there is an even number of ‘ones’.

When creating the message $r_{ji}(0)$ which is to be sent back to variable node χ_i , the check node determines the probability that there is an even number of ‘ones’ in the other associated bits ($R_{j/i}$). Likewise when creating $r_{ji}(1)$, it determines the probability that there is an odd number of ‘ones’. It can be then shown that [21]:

$$\begin{aligned}
 r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{j/i}} (1 - 2P_{i'j}) \\
 r_{ji}(1) &= \frac{1}{2} - \frac{1}{2} \prod_{i' \in R_{j/i}} (1 - 2P_{i'j})
 \end{aligned} \tag{2.1}$$

and accordingly that

$$\begin{aligned} q_{ij}(0) &= (1 - P_i) \prod_{j' \in C_{i/j}} r_{j'i}(0) \\ q_{ij}(1) &= \prod_{j' \in C_{i/j}} r_{j'i}(1) \end{aligned} \tag{2.2}$$

Thus, the algorithm works as follows: [21]

1. Initialization:

P_i , as well as $q_{ij}(0)$ and $q_{ij}(1)$ are received from the demapper based on what type of modulation scheme is used.

2. Computation of r_{ji} :

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{j/i}} (1 - 2q_{i'j}(1)) \\ r_{ji}(1) &= 1 - r_{ji}(0) \end{aligned} \tag{2.3}$$

3. Computation of q_{ij} :

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1 - P_i) \prod_{j' \in C_{i/j}} r_{j'i}(0) \\ q_{ij}(1) &= K_{ij}P_i \prod_{j' \in C_{i/j}} r_{j'i}(1) \end{aligned}$$

where K_{ij} is a constant used to ensure that $q_{ij}(0) + q_{ij}(1) = 1$.

4. Computation of Q_i :

For all i ,

$$Q_i(0) = K_i(1 - P_i) \prod_{j \in C_i} r_{ji}(0)$$

$$Q_i(1) = K_i P_i \prod_{j \in C_i} r_{ji}(1)$$

where K_i is a used constant to ensure that $Q_i(0) + Q_i(1) = 1$.

5. Computation of the current best estimates for c_i , denoted as $\hat{c}_i$:

$$\hat{c}_i = \begin{cases} 0 & \text{if } Q_i(0) > 0.5 \\ 1 & \text{otherwise} \end{cases}$$

If the new $\hat{\mathbf{c}}$ satisfies all check equations or the maximum number of iterations is reached; Stop. Else, return to Step 2.

Use of the above algorithm is computationally intense, due to the large number of multiplications. It is also numerically unstable, due to the sheer size of the numbers involved. Thus to improve performance and speed, a Log Domain version is commonly used. This also reduces the number of messages that need to be sent along the edges of the graph. Instead of sending $c_i(0)$ and $c_i(1)$, the *Log-Likelihood Ratio*(LLR) of c_i is sent, where the LLR of c_i is:

$$\text{LLR}(c_i) = \log_e \frac{Pr(c_i = 0|y_i)}{Pr(c_i = 1|y_i)} \quad (2.4)$$

Similarly the following LLRs are used as well:

$$\begin{aligned}\text{LLR}(r_{ji}) &= \log_e \frac{r_{ji}(0)}{r_{ji}(1)} \\ \text{LLR}(q_{ij}) &= \log_e \frac{q_{ij}(0)}{q_{ij}(1)} \\ \text{LLR}(Q_i) &= \log_e \frac{Q_i(0)}{Q_i(1)}\end{aligned}$$

The demapper will pass the LDPC decoder $\text{LLR}(c_i)$ and the decoder will use $\text{LLR}(Q_i)$ to determine if a bit is a ‘one’ or a ‘zero’. If $\text{LLR}(Q_i)$ is positive it will be decoded as a ‘zero’, if it is negative it will be decoded as a ‘one’.

There are many more speed and simplicity optimizations that can be used in the Log Domain. These other optimizations are very well detailed in many papers and textbooks [21, 24, 25] and will not be discussed further in this document.

2.2 Multi-Level Coding

Multi-Level Coding (MLC) [10, 11] was introduced in May 1977 and has since become one of the most well known coded modulation schemes for efficient communication over power and bandwidth limited channels. MLC requires m encoders each of which outputs one coded bit $c_i, i = 0, 1, \dots, m - 1$ per symbol. Each encoder is of length n and rate $R_i = k_i/n$, where k_i is the length of the information word. Since each encoder can have a different rate, each encoder can also have a different number of input bits, such that the *output* codeword length is the same for all encoders. The coded bits, $\mathbf{c} = (c_0, \dots, c_{m-1})$, are then mapped onto a symbol and sent across the

channel. This is illustrated in Figure 2.3 [12].

Figure 2.3: MLC Encoder

Optimal performance of MLC comes when each bit is equally protected [11, 12], meaning the total protection offered by coding and by modulation is the same for each bit. Since the rates of the component codes vary, each bit might not be equally protected by coding. Likewise the Euclidean distance between symbols vary as well, thus each bit might not be equally protected by modulation. A balance must be struck that attempts to have each bit overall equally protected.

2.2.1 Multi-Stage Decoding

Multi-stage Decoding (MSD) is a method of decoding MLC. MSD uses the results of the previous stages to better estimate the results of the current stage. Thus each component code, C^i , is decoded based on the output of the previous codes [12].

In the following, upper case variables will correspond to random variables associated to the transmitted and received symbols. Assuming i.i.d equiprobable inputs,

the capacity of the channel, with x as the input and y as the output is given by [12]:

$$C = I(C_0, C_1, \dots, C_{m-1}; Y) = I(X; Y) \quad (2.5)$$

Using the mutual information chain rule on (2.5) it can be seen that [12]:

$$\begin{aligned} I(C_0, C_1, \dots, C_{m-1}; Y) &= I(C_0; Y) + I(C_1; Y|C_0) + \dots \\ &\quad + I(C_{m-1}; Y|C_0, \dots, C_{m-2}) \end{aligned} \quad (2.6)$$

From this it can be deduced that the vector $\mathbf{c} = (c_0, \dots, c_{m-1})$ can be decoded sequentially, first by decoding the lowest level and then passing the output to the next level, continuing until all bits are decoded. This process is shown in Figure 2.4

2.2.2 Parallel Independent Decoding

Parallel Independent Decoding (PID) is a sub-optimal decoding strategy yet is still very effective. It has the advantage of being less complex and having a shorter delay, when compared to MSD. It is very similar to MSD decoding except that it does not take into account the results from the previous stages. This means that each code can be decoded in parallel, instead of waiting until the previous code was done. This means a shorter delay, and less complexity since the decoders do not all need to be interconnected.

Although all c_i 's in vector $\mathbf{c} = (c_0, \dots, c_{m-1})$ are originally independent of each other before being mapped to the constellation symbols, this is *not* the case after mapping. Thus the following can be derived,

Figure 2.4: MSD Decoder

$$\sum_{i=0}^{m-1} I(C_i; Y) \leq I(C_0, C_1, \dots, C_{m-1}; Y) \quad (2.7)$$

It is shown in [11,12,14] that the gap between the two sides of the inequality strongly depends on the type of mapping scheme used, and that if Gray mapping is used this gap is quite small. PID takes advantage of this small loss in optimality and decodes each bit independently. Since each decoder does not need to take into account the results from the previous stage, this reduces the delay by a factor of m and also reduces the complexity. This is shown in Figure 2.5.

Figure 2.5: PID Decoder

2.2.3 MLC and LDPC

Optimal performance of MLC occurs when the *overall protection* for each bit is the same. Thus bits that are closer together in the signal constellation must be more

protected by coding [11]. MLC varies the protection offered to each bit by adjusting the component code rates. LDPC codes are extremely flexible with respect to rate, virtually any rate desired can be generated by choosing the appropriate (n, m, k) combination.

Jilei Hou *et al* showed in [12] how LDPC code rate could be optimized for various modulation schemes to improve performance.

2.3 Bit-Interleaved Coded Modulation

Bit-Interleaved Coded Modulation (BICM) [14] was proposed by Zehavi [13] to improve performance over Rayleigh fading channels. The idea is a simple but effective approach for bandwidth efficient transmission. It requires only one encoder, whose coded output is bit-wise interleaved and grouped into m address bits, $c_0, c_1, \dots, c_{m-1}$. The signal corresponding to address $c_0, c_1, \dots, c_{m-1}$ is then sent across the channel. The received vector is sent to the demapper, which computes the *a priori* information for the decoder. These are de-interleaved and passed to the decoder. This is illustrated in Figure 2.6.

As one can see BICM is a very simple scheme. It was originally designed to improve performance over Rayleigh fading channels, but is now used over AWGN channels as well. The random interleaver, however, makes the optimization of BICM difficult. It means that little can be done at the encoder stage since the bits will be randomized before being passed to the signal mapper. This is referred to as “random modulation” [13]. It should be stressed that this is *intentional*, that BICM deliberately forces coding and modulation to be separate. Instead of optimizing each code for every possible channel, codes are instead only optimized for a given channel

Figure 2.6: BICM Encoder & Decoder

(AWGN, Rayleigh Fading, etc). The random interleaver then transforms the actual channel to resemble the given channel.

2.3.1 BICM and LDPC Codes

Unlike MLC there seems to be very little linkage between modulation and LDPC-based coding in using BICM (though this document aims to change this!). BICM is largely independent of the type of coding used. Modulation and coding are separately optimized and put together. In most cases a different encoder/decoder pair can be substituted, so long as the codeword length and rate were the same.

However, when using LDPC codes there can be a small saving in complexity. Due to the way LDPC codes are created from a parity-check matrix, instead of putting in an interleaver one can simply randomize the matrix. This has the same net effect of interleaving the source bits as compared to the output bits but saves on complexity [26]. This is assuming a relatively fast fading channel, if the fade were as long or longer than the codeword bit interleaving would not be sufficient.

2.4 MLC and BICM

MLC and BICM are both well studied coded modulation schemes, and there has been a body of work on their respective performances when used with LDPC codes [1,12,26,27]. It was shown in [12] that the performance of optimized MLC and BICM at short to moderate block lengths are very similar. However since MLC requires many encoders, and more delay as compared to BICM, the following conclusion was reached [12]:

At small to moderate block size, BICM seems to be the more attractive alternative, achieving a better balance among complexity, delay, and performance.

Based on this conclusion, that although extensive optimization of the component code rates for MLC had been done, it still was a less attractive alternative than BICM, it was decided that the focus of this research would be on ways to improve the performance of BICM.

2.5 Channel Adapters

Many useful tools in coding, such as *density evolution*, are greatly simplified if the channel is symmetric. Unfortunately, for MLC and BICM the equivalent binary channels are, in general, *non-symmetric* [12]. This also means that using the all-zero codeword exclusively may not accurately reflect the true average. A binary-input channel is said to be symmetric if

$$p(Y = y|C = 0) = p(Y = -y|C = 1) \quad (2.8)$$

where C is the input and Y is the output of the binary-input channel [28].

To overcome this limitation, [12] introduces a new analytical tool: i.i.d Channel Adapters. They create equivalent binary-input channels that are provably symmetric, and thus allow the all-zero codeword to be used exclusively and will reflect the actual average performance.

A Channel Adapter consists of three parts: firstly an i.i.d. source which generates $t_i \in \{0, 1\}$ with equal probability. The second part is a modulo-2 adder, which performs $d_i = c_i \oplus t_i$. The third and last part adjusts the sign of the input, as follows [12]:

$$v_i = \begin{cases} u_i & t_i = 0 \\ -u_i & t_i = 1 \end{cases}$$

The Channel Adapter generates a random bit t_i and XORs it with the coded bit c_i to create d_i . It is then passed to the signal mapper and then across the channel. The symbol is demapped and the resulting LLRs are sent to the last part of the Channel Adapter. If the bit was modified (i.e. if $t_i = 1$) the sign of the LLR is reversed. [12]

This simple tool ensures that no matter what the input sequence is, a ‘random’ sequence of bits is fed into the signal mapper, but after demapping the LLRs are adjusted to reflect the original input sequence. Channel adapters allow the use of the ‘all-zero’ codeword, even though the channel might not be symmetric.

Chapter 3

Modulation Schemes

Modulation is the act of transforming information into a signal that can be transmitted across a channel. Two broad forms of modulation are used in this document: *Pulse Amplitude Modulation (PAM)* and *Phase Shift Keying (PSK)*. Both are very well known and mature modulation schemes.

It is important to know a great deal about the particular *Signal Constellation* used, as well as the corresponding *Signal Mapping*. The Signal Constellation is the exact arrangement of points in the signal space. For example, the 8-PSK signal constellation has eight signal points uniformly distributed around a unit circle, whereas 64-PSK has sixty-four points. Some schemes, like Quadrature Amplitude Modulation (QAM), arrange signal points in rectangular grids. There are many different possibilities and each can yield very different results. Signal Mapping, also called Signal Labelling, is used to determine which sequence of bits correspond to each signal point. Gray Mapping (GM) is the most common scheme and is used almost exclusively in this work.

In most non-binary modulation schemes each bit is not equally protected. This

arises since the minimum Euclidean distance (d_{min}) for each bit, meaning the Euclidean distance between a symbol and the closest symbol whose bit has a different value, are different. As a result of this some bits are more protected than others. This can be shown empirically using a histogram of the LLRs produced by the decoder, using a different curve for each bit. Channel Adapters are used since modulation schemes are not always symmetric, as mentioned in Section 2.5. It is desirable to determine which bits are better protected, and to what degree. To this end the LLR is measured after the Channel Adapter, just before it is fed into the LDPC decoder, as shown in Figure 3.1. This allows us to determine at a glance the bits that were received in error. Ideally all LLRs should be positive, and a negative value indicates an error. If the LLRs were measured before the Channel Adapter it is difficult to identify erroneous bits, since there would be many valid positive and negative LLRs. The histogram of the LLRs will be shown and explained for each modulation scheme, using various E_b/N_0 , in the following sections¹.

Figure 3.1: LDPC Decoder and Channel Adapter

¹For all histograms, there is always one hundred points for each curve. Thus the range for each ‘bin’ is (total range)/100.

3.1 4-PAM

Pulse Amplitude Modulation is a very simple modulation scheme. Instead of simply sending ‘+1’ and ‘-1’ to represent binary digits, PAM uses many levels to represent many binary digits. In fact, simple bipolar modulation can be viewed as a special case of PAM where the number of levels is two.

4-PAM has four levels, each evenly spaced along one dimension. It thus has a spectral efficiency of 2 bits/dimension. The average energy per symbol of this 4-PAM scheme is $E_s = [(-3)^2 + (-1)^2 + 1^2 + 3^2] / (4) = 5$ and therefore $E_b = 2.5$. Figure 3.2 shows the 4-PAM Gray Mapped modulation scheme used in this document. Signal points in the figure, and in all future figures, are labelled with a string of bits: $\{c_0 c_1 c_2 \dots c_{M-1}\}$. Therefore the most significant bit is c_0 and the least significant bit is c_{M-1} .

Figure 3.2: Signal Constellation for 4-PAM Gray Mapping

3.1.1 LLR Analysis

Graphing a histogram of the LLRs that are passed to the LDPC decoder shows empirically the performance difference between the two bits of a 4-PAM symbol. Figure 3.3 is a collection of five different LLR histograms, for E_b/N_0 0, 2, 5, 10 and 20 dB² used to illustrate the different protection offered to each bit. The histogram

²Unless explicitly stated all E_b/N_0 are in dB.

was generated using 5 000 000 bits. In addition, to further show the difference in protection, two tables have also been included: Table 3.1 gives the mean LLR values, and Table 3.2 gives the probability that the LLR is below zero. The latter statistic is very useful, as it indicates the probability that a bit will be received in error.

$E_b/N_0(\text{dB})$	Mean of LLR	
	c_0	c_1
0	4.3859	1.4777
2	6.6370	2.4383
5	12.7885	5.0062
10	40.0034	15.9949
20	464.6704	159.9929

Table 3.1: Mean of LLR, 4-PAM Gray Mapping

$E_b/N_0(\text{dB})$	P(LLR)<0	
	c_0	c_1
0	0.0944	0.1872
2	0.0650	0.1302
5	0.0277	0.0560
10	0.0012	0.0023
20	0.0000	0.0000

Table 3.2: Probability that LLR<0, for 4-PAM Gray Mapping

Figure 3.3 illustrates two trends: firstly the mean of the most significant bit, denoted in the graph as ‘bit 0’ (c_0), is higher than ‘bit 1’ (c_1) for all E_b/N_0 , and secondly that the area beneath the curve for which LLR<0 is smaller for c_0 than for c_1 . These trends show that c_0 is better protected than c_1 under 4-PAM modulation.

It is worth addressing two additional points of interest about Figure 3.3: firstly LLR(c_1) exhibits a spike for E_b/N_0 0, 2 & 5, and secondly LLR(c_0) shows two distinct crests. These crests become more pronounced for higher E_b/N_0 . The spike

in the $\text{LLR}(c_1)$ is due to the fact that the $\text{LLR}'(c_1)$ (the LLR just before the Channel Adapter) is lower bounded. The equation for calculating $\text{LLR}'(c_1)$ is:

$$\begin{aligned}\text{LLR}'(c_1) &= \log_e \left[\frac{p(c_1 = 0|y)}{p(c_1 = 1|y)} \right] \\ &= \log_e \left[\frac{e^{-\frac{(y-3)^2}{2\sigma^2}} + e^{-\frac{(y+3)^2}{2\sigma^2}}}{e^{-\frac{(y-1)^2}{2\sigma^2}} + e^{-\frac{(y+1)^2}{2\sigma^2}}} \right]\end{aligned}\tag{3.1}$$

The minimum of $\text{LLR}'(c_1)$ occurs when $y = 0$, thus using (3.1)

$$\begin{aligned}\min(\text{LLR}'(c_1)) &= \log_e \left[\frac{e^{-\frac{9}{2\sigma^2}} + e^{-\frac{9}{2\sigma^2}}}{e^{-\frac{1}{2\sigma^2}} + e^{-\frac{1}{2\sigma^2}}} \right] \\ &= \log_e \left[\frac{e^{-\frac{9}{2\sigma^2}}}{e^{-\frac{1}{2\sigma^2}}} \right] \\ &= \log_e \left[e^{-\frac{9}{2\sigma^2}} \right] - \log_e \left[e^{-\frac{1}{2\sigma^2}} \right] \\ &= -\frac{9}{2\sigma^2} - \left(-\frac{1}{2\sigma^2} \right) \\ &= -\frac{4}{\sigma^2}; \quad \text{where } \sigma^2 = \frac{N_0}{2}, \text{ and } E_b = 2.5 \\ &= -\frac{E_b}{2.5} \cdot \frac{8}{N_0} \\ &= -3.2 \cdot \frac{E_b}{N_0} \text{ [linear]}\end{aligned}\tag{3.2}$$

Equation 3.2 shows that $\text{LLR}'(c_1)$ is lower bounded by a negative value. After passing through the Channel Adapter all correct values are positive while all erroneous values are negative. The congregation of points around the negative lower bound translates into the small spike seen in the graphs. As shown in Table 3.3 the values closely match the spikes present in Figure 3.3. No spikes are seen when the E_b/N_0 is equal to 10 dB or 20 dB (Figures 3.3(d) and 3.3(e)) because the variance of the AWGN is

so small at high E_b/N_0 that it is very unlikely to $y = 0$, and thus the spike, to occur.

E_b/N_0 [dB]	E_b/N_0 [Linear]	$ \min(\text{LLR}'(c_1)) $
0	1	3.2
2	1.5849	5.0717
5	3.1623	10.1193
10	10	32
20	100	320

Table 3.3: Absolute value of lower limit of $\text{LLR}'(c_1)$

As the E_b/N_0 increases, $\text{LLR}(c_0)$ shows two crests, each becoming more pronounced as E_b/N_0 increases. Since 4-PAM is symmetric around zero, there are only two distinct cases, either $x = \pm 1$ or $x = \pm 3$. Let s_1 represent $x = \pm 1$ and let s_3 represent $x = \pm 3$.

The right hand crest represents bits decoded when s_3 was sent, and the left hand crest represents bits decoded when s_1 was sent. This is best explained using the Distance Spectrum, as shown in Table 3.4. A similar table was used in [1] and has been adapted to 4-PAM. Following the notation used in [1], d_1^i is denoted as the distance to another symbol with a different bit in position i . d is the distance between two neighbouring signal points, and in this case $d = 2$.

Bit c_i	Symbol	d_1^i
c_0	s_3	$\{2d, 3d\}$
c_0	s_1	$\{d, 2d\}$
c_1	s_1	$\{d, 2d\}$
c_1	s_3	$\{d, 2d\}$

Table 3.4: Distance Spectrum for 4-PAM (GM)

Looking at Table 3.4 it is clear that c_0 when s_3 is sent, has a larger $d_{\min}$ than the other three cases. The $d_{\min}$ of c_0 is $2d$ when s_3 is sent but only d when s_1 is sent. The

d_{min} of c_1 is always d regardless of the signal sent. Since the performance is heavily based on d_{min} it follows that the two crests seen in $\text{LLR}(c_0)$ correspond to whether s_1 or s_3 was sent (s_1 for the left hand curve, s_3 for the right). It also explains why the left-most crest closely matches the curve for c_1 , since they have the same d_{min} . To further show this we can use equation for calculating $\text{LLR}'(c_i)$:

$$\begin{aligned}\text{LLR}'(c_0) &= \log_e \left[\frac{p(c_0 = 0|y)}{p(c_0 = 1|y)} \right] \\ &= \log_e \left[\frac{e^{-\frac{(y-3)^2}{2\sigma^2}} + e^{-\frac{(y-1)^2}{2\sigma^2}}}{e^{-\frac{(y+3)^2}{2\sigma^2}} + e^{-\frac{(y+1)^2}{2\sigma^2}}} \right]\end{aligned}\tag{3.3}$$

$$\begin{aligned}\text{LLR}'(c_1) &= \log_e \left[\frac{p(c_1 = 0|y)}{p(c_1 = 1|y)} \right] \\ &= \log_e \left[\frac{e^{-\frac{(y-3)^2}{2\sigma^2}} + e^{-\frac{(y+3)^2}{2\sigma^2}}}{e^{-\frac{(y-1)^2}{2\sigma^2}} + e^{-\frac{(y+1)^2}{2\sigma^2}}} \right]\end{aligned}\tag{3.4}$$

Using (3.3) and (3.4) the LLRs for when symbols s_3 and s_1 are perfectly received (received without noise) can be calculated. These values correspond to the ‘peaks’ seen in the histograms in Figures 3.3(a), 3.3(b) and 3.3(c). Table 3.5 shows the values of the $\text{LLR}(c_i)$ ’s for various E_b/N_0 and for different symbols. They are based on the above equations, though are all positive due to the channel adapter (Section 2.5)

These values confirm that the right-hand peak occurs when s_3 is sent, and the left-hand when s_1 is sent. They also show that the peaks for c_0 and for c_1 are equal when s_1 is sent. This is expected since, from the distance spectrum, they have the same d_{min} .

$E_b/N_0(\text{dB})$	Symbol	LLR(c_0)	LLR(c_1)
0	s_1	1.775705	1.775705
	s_3	6.583565	1.591804
2	s_1	2.611549	2.611549
	s_3	10.219530	2.535333
5	s_1	5.065972	5.065972
	s_3	20.244905	5.059644
10	s_1	16	16
	s_3	64	16
20	s_1	160	160
	s_3	640	160

Table 3.5: Peak values of LLR, 4-PAM Gray Mapping

(a) $E_b/N_0 = 0$ dB (b) $E_b/N_0 = 2$ dB

Figure 3.3: Histogram of LLRs for 4-PAM Modulation, with Gray Mapping

(c) $E_b/N_0 = 5$ dB (d) $E_b/N_0 = 10$ dBFigure 3.3: *cont.* Histogram of LLRs for 4-PAM Modulation, with Gray Mapping

(e) $E_b/N_0 = 20$ dBFigure 3.3: *cont.* Histogram of LLRs for 4-PAM Modulation, with Gray Mapping

3.2 8-PSK

Phase Shift Keying is another very well known modulation scheme. Signal points are arranged around a circle, usually (though not necessarily) evenly spaced. This particular 8-PSK scheme has eight evenly spaced signal points around a unit circle, as shown in Figure 3.4³. It has three bits per symbol, and thus a spectral efficiency of 1.5 bits/dimension. All points are on the unit circle, thus $E_s = 1$ and $E_b = 1/3$.

3.2.1 LLR Analysis

The histograms of the LLRs are shown in Figure 3.5, and the accompanying Tables 3.6 and 3.7 show the mean and probability of error for the LLRs, respectively. The histogram was generated using 500 000 bits. The performance of c_0 and c_1 are nearly identical, while the performance of c_2 is considerably lower. This bimodal performance, as with 4-PAM, is due to the difference in protection offered by the modulation scheme.

$E_b/N_0(\text{dB})$	Mean of LLR		
	c_0	c_1	c_2
0	4.1571	4.1473	1.5960
2	6.3664	6.3535	2.6640
5	12.3644	12.3483	5.4962
10	38.7989	38.7725	17.5660
20	390.0499	390.2430	175.7180

Table 3.6: Mean of LLR, 8-PSK Gray Mapping

The difference in protection can be explained by first looking at Figure 3.4, where the signal constellation is broken up into three subset signal constellations. For c_2 ,

³This diagram is slightly modified version of a diagram by Ravi Narayanaswami, used in [1]

Figure 3.4: Signal Constellation for 8-PSK Gray Mapping [1]

$E_b/N_0(\text{dB})$	$P(\text{LLR}) < 0$		
	c_0	c_1	c_2
0	0.0931	0.0928	0.1806
2	0.0612	0.0598	0.1200
5	0.0243	0.0234	0.0478
10	6.9048×10^{-4}	7.0238×10^{-4}	0.0015
20	0.0000	0.0000	0.0000

Table 3.7: Probability that $\text{LLR} < 0$, for 8-PSK Gray Mapping

for any of the signals, one of its nearest neighbours is always different, and one always the same. Bits c_0 and c_1 have two cases: either the signal has one nearest neighbour which is different and one which is the same, denoted as an *Edge Symbol* (E), or it has both which are the same, denoted as an *Inner Symbol* (I) [1]. This is shown concisely in Table 3.8, showing the distance spectrum for each bit. Following notation used in [1], d_1^i denotes the distance to another symbol with a different bit in position i .

Bit c_i	Symbol	d_1^i
c_0	Inner Symbol	$\{d_1, d_2, d_2, d_3\}$
c_0	Edge Symbol	$\{d_0, d_1, d_2, d_3\}$
c_1	Inner Symbol	$\{d_1, d_2, d_2, d_3\}$
c_1	Edge Symbol	$\{d_0, d_1, d_2, d_3\}$
c_2	-	$\{d_0, d_1, d_1, d_2\}$

Table 3.8: Distance Spectrum for 8-PSK (GM) [1]

The distances in Figure 3.8 represent all the distances to symbols who have a different bit in position i . For example, examine symbol ‘000’ for $i = 0$. To complete the distance spectrum one must locate all symbols where $c_0 = 1$, and then find the distance from ‘000’ to each of those symbols. In this case the symbols where $c_0 = 1$ are $\{100, 101, 111, 110\}$, and the distances to those symbols are $\{d_0, d_1, d_2, d_3\}$. The distance spectrum can be used to examine the protection offered by modulation to different symbols. The minimum value in the distance spectrum is also d_{min} .

The distance spectrum helps to explain some characteristics about the curves in Figure 3.5. Firstly, two distinct crests can be seen in the curves for c_0 and c_1 for high E_b/N_0 . Secondly, the shape of the left hand histograms are very similar at high E_b/N_0 , although c_2 is twice the height. The right hand crest represents an *Inner Symbol*, and the left hand crests represents an *Edge Symbol*. Performance is dominated by d_{min} , and thus looking at the distance spectrum of either c_0 or c_1 , the

d_{min} for an Inner Symbol is d_1 whereas for an Edge Symbol it is d_0 . Since $d_1 > d_0$ it follows that the average LLR for an Inner Symbol will be greater than that of an Edge Symbol. The curves for c_2 and an Edge Symbol are very similar since they have the same $d_{min} = d_0$.

It should be noted that, as observed in [1], every signal point contains exactly one Edge Symbol, one Inner Symbol and c_2 . Based on this, it could be argued that the *all-zero codeword* could be used exclusively [1]. This will indeed cause a similar number of bits to be received in error, but it will mean that c_1 will always be an Inner Symbol, and c_0 will always be an Edge Symbol. This is generally not of great importance to most schemes, however to avoid it, Channel Adapters have been used.

(a) $E_b/N_0 = 0$ dB (b) $E_b/N_0 = 2$ dB

Figure 3.5: Histogram of LLRs for 8-PSK Modulation, with Gray Mapping

(c) $E_b/N_0 = 5$ dB (d) $E_b/N_0 = 10$ dBFigure 3.5: *cont.* Histogram of LLRs for 8-PSK Modulation, with Gray Mapping

(e) $E_b/N_0 = 20$ dBFigure 3.5: *cont.* Histogram of LLRs for 8-PSK Modulation, with Gray Mapping

3.3 16-PSK

This scheme is similar to 8-PSK, except that it has 16 symbols. It has four bits per symbol, and therefore has a spectral efficiency of 2 bits/dimension. All the points are arranged on the a unit circle, at an angle of $\alpha \cdot \frac{\pi}{8}$, $\alpha = \{0, 1, 2, \dots, 15\}$. Therefore $E_s = 1$ and $E_b = 1/4$. A diagram of 16-PSK is similar to 8-PSK but with 16 symbols. A list of signal labels for Gray Mapping are given in Appendix A.1

3.3.1 LLR Analysis

The analysis of 16-PSK is very similar to those of the previous two sections. Figure 3.6 shows a histogram of the LLRs and Tables 3.9 and 3.10 show the mean and probability of error respectively. The histogram was generated using 500 000 bits. The most reliable bit is the most significant bit, c_0 , the second most reliable bit is c_1 and so on until c_3 , which is the least reliable bit. This is shown in Table 3.9 where the means of the LLRs are as follows: $c_0 \cong c_1 > c_2 > c_3$, for $E_b/N_0 = 0, 2, 5, 10, 20$.

$E_b/N_0(\text{dB})$	Mean of LLR			
	c_0	c_1	c_2	c_3
0	4.6255	4.6504	1.6312	0.3238
2	6.8814	6.9217	2.5232	0.6934
5	12.9368	13.0145	4.7833	1.7218
10	39.6643	39.9094	14.6426	6.0235
20	2.6320×10^3	2.4882×10^3	145.9541	60.9124

Table 3.9: Mean of LLR, 16-PSK Gray Mapping

16-PSK follows the same trend as in 8-PSK, where the most significant bits are more protected than the least significant ones. The performance of c_0 is equal to that of c_1 , since each sub-constellation comprises one half of the unit circle. c_0 can

E_b/N_0 (dB)	P(LLR)<0			
	c_0	c_1	c_2	c_3
0	0.0878	0.0914	0.1815	0.3378
2	0.0646	0.0691	0.1369	0.2647
5	0.0395	0.0420	0.0847	0.1641
10	0.0092	0.0103	0.0202	0.0401
20	0.0000	0.0000	0.0000	0.0000

Table 3.10: Probability that LLR<0, for 16-PSK Gray Mapping

be determined from the upper and lower half of the circle, while c_1 from the left and right halves.

Figure 3.6 shows that crests appear as E_b/N_0 increases. There are more crests than for 8-PSK, which is to be expected. The reason for the crests is the same as for 8-PSK, except that 16-PSK has more classes of symbols. c_3 has only one class, c_2 has two classes, and c_1, c_0 have four classes.

(a) $E_b/N_0 = 0$ dB (b) $E_b/N_0 = 2$ dB

Figure 3.6: Histogram of LLRs for 16-PSK Modulation, with Gray Mapping

(c) $E_b/N_0 = 5$ dB (d) $E_b/N_0 = 10$ dBFigure 3.6: *cont.* Histogram of LLRs for 16-PSK Modulation, with Gray Mapping

(e) $E_b/N_0 = 20$ dBFigure 3.6: *cont.* Histogram of LLRs for 16-PSK Modulation, with Gray Mapping

3.4 64-PSK

This scheme is similar to previous PSK schemes, except that it has 64 symbols. It has six bits per symbol, and therefore has a spectral efficiency of 3 bits/dimension. Again, all points are on the unit circle, at an angle of $\alpha \cdot \frac{\pi}{32}$, $\alpha = \{0, 1, 2, \dots, 63\}$. Therefore $E_s = 1$ and $E_b = 1/6$. A list of signal labels for Gray Mappings are given in Appendix A.2.

3.4.1 LLR Analysis

The analysis of 64-PSK is very similar to those of the previous three sections. Figure 3.7 shows a histogram of the LLRs and Tables 3.11 and 3.12 show the mean and probability of error respectively. The histogram was generated using 500 000 bits. The most reliable bit is the most significant bit, c_0 , the second most reliable bit is c_1 and so on until c_5 , which is the least reliable bit.

$E_b/N_0(\text{dB})$	Mean of LLR					
	c_0	c_1	c_2	c_3	c_4	c_5
0	6.2523	5.8697	2.1470	0.5185	0.0160	-7.5959×10^{-6}
2	9.2093	8.6060	3.1422	0.9426	0.0709	-1.6894×10^{-5}
5	16.9831	15.7736	5.5372	1.8857	0.3478	0.0037
10	50.5844	46.6852	15.3882	5.0613	1.5647	0.2574
20	3284.8	2900.4	147.0173	45.6967	14.3737	5.6886

Table 3.11: Mean of LLR, 64-PSK Gray Mapping

The 64-PSK scheme shows the same trend of differing reliabilities but it is more pronounced. Table 3.12 shows that the probability of error for c_0 is *seven to sixteen times lower* than that of c_5 , for $E_b/N_0 = \{0, 2, 5, 10, 20\}$. This result, as will be discussed later, is quite significant. The increased difference is due to the increased

$E_b/N_0(\text{dB})$	$P(\text{LLR}) < 0$					
	c_0	c_1	c_2	c_3	c_4	c_5
0	0.0708	0.0861	0.1517	0.2874	0.4639	0.5028
2	0.0561	0.0697	0.1201	0.2233	0.4210	0.4992
5	0.0394	0.0508	0.0851	0.1501	0.3229	0.4820
10	0.0212	0.0288	0.0474	0.0776	0.1773	0.3438
20	0.0029	0.0037	0.0062	0.0101	0.0215	0.0448

Table 3.12: Probability that $\text{LLR} < 0$, for 64-PSK Gray Mapping

difference in $d_{\min}$.

Figure 3.7 is more complicated in this case than in previous ones. Crests begin to appear at higher E_b/N_0 . This is caused by the same phenomenon discussed in Section 3.2.1, though instead of just Inner and Edge symbols, there are many types of symbols for different bits. There is a crest corresponding to each different type of symbol.

(a) $E_b/N_0 = 0$ dB (b) $E_b/N_0 = 2$ dB

Figure 3.7: Histogram of LLRs for 64-PSK Modulation, with Gray Mapping

(c) $E_b/N_0 = 5$ dB (d) $E_b/N_0 = 10$ dBFigure 3.7: *cont.* Histogram of LLRs for 64-PSK Modulation, with Gray Mapping

(e) $E_b/N_0 = 20$ dBFigure 3.7: *cont.* Histogram of LLRs for 64-PSK Modulation, with Gray Mapping

Chapter 4

Reliability-Based Coded Modulation

It has been shown in the previous chapter that non-binary modulation schemes do not equally protect each bit in a symbol. This difference in protection is not trivial. In some cases certain bits are substantially more likely to be received in error than others. In theory, there should be a way to use this knowledge; a way to design a scheme that takes advantage of the known difference in protection. Multi-Level Coding (MLC), as discussed in Section 2.2, recognizes this and attempts to equally protect each bit by adjusting the rates of the component codes [12]. Reliability-Based Coded Modulation (RBCM) is a method for putting this theory into practice for BICM. It works by interchanging the columns of the parity-check matrix in a very specific manner, depending on the modulation scheme and LDPC code used. Since the only change is a one-time modification to the parity-check matrix, it does not add *any additional complexity or delay* to the system.

There are two different versions of RBCM that will be considered: *Variable Node*

Optimization (VNO) and *Check Node Optimization* (CNO). VNO was conceived first and ultimately failed to significantly improve performance¹. CNO was developed later and can be shown to give considerable gains for regular LDPC codes, provided that the modulation scheme and LDPC code were properly chosen.

4.1 Variable Node Optimization

Variable Node Optimization (VNO) is only possible for irregular LDPC codes. With irregular codes, just as all bits are not equally protected by modulation, all bits are also not equally protected by coding. It can be shown, that variable nodes with high degrees (d_v) are better protected. This is true since the higher the d_v , the more check-nodes they are associated with, and thus the more messages they receive. With more messages a variable node is better able to arrive at its correct state.

VNO attempts to couple the known unequal modulation protection with the known unequal coding protection. Permuting the columns of the parity-check matrix controls which bit is assigned to each variable node. Next VNO decides which bits should be mapped to which nodes. This turns out to be equivocal. If bits which have been deemed ‘unreliable’ (meaning the least protected bit of a given symbol) are mapped to variable nodes with high degree, their chances of converging to the correct state is higher. However, while nodes with high degree receive many messages, they also *send out* many messages. Thus if unreliable bits are mapped to high degree nodes, their unreliable information will be widely spread across the graph. Alternatively, the most reliable bits can be mapped to nodes with high degree. This means that their highly reliable information will be spread across the graph, while the effect

¹This will be shown in Chapter 5.

of the error-prone information from the unreliable nodes will be limited. This also means that the unreliable bits will be mapped to low degree nodes, unfortunately causing the bits least protected by modulation to also be the bits least protected by coding.

The VNO tradeoff can be viewed as follows: if reliable bits are mapped to high degree nodes, denoted as VNO1, then the system will have higher *initial* performance but lower *iterative* performance. If instead unreliable bits are mapped to high degree nodes, denoted as VNO2, the system will have lower *initial* performance but higher *iterative* performance. Accordingly, VNO1 outperforms VNO2 and in some cases shows slight improvement over random interleaving for low E_b/N_0 , while VNO2 always shows inferior performance to BICM. For very high E_b/N_0 the curves appear to cross, with VNO2 overtaking VNO1 in performance. Thus neither case proved to be significantly better than random interleaving, and for high E_b/N_0 both cases show inferior performance².

4.1.1 Implementation

VNO can be implemented quite easily. As a simple scenario, for any of the modulation schemes discussed in Chapter 3 one in M bits will be deemed unreliable, where M is the number of bits per symbol. It is thus a simple matter to ensure that every M^{th} column is either a high weight or a low weight column, depending on the type of performance required.

²See Chapter 5.

4.2 Check Node Optimization

Check Node Optimization (CNO) works best on regular LDPC codes. It can still function on irregular LDPC codes, though it does not generate a significant improvement. CNO attempts to organize the parity-check matrix such that the unreliable bits are evenly distributed among the check nodes. Preferably they are distributed such that one, and only one, unreliable bit is associated to each check node. This is preferable because if a check node receives more than one unreliable message, it greatly reduces its capacity to help the associated variable nodes reach their correct value. If a check node has more than one message with $\text{LLR} = 0$, all its outgoing LLRs will be zero.

In order to better understand how check nodes cope with very unreliable messages, one can consider the following scenario. Bit c_I is so poorly received that it is equally likely that its a ‘zero’ or a ‘one’,

$$Pr(c_I = 0|y) = Pr(c_I = 1|y) = 0.5$$

or in the log domain $\text{LLR}(c_I) = 0$. Thus messages $q_{Ij}(0) = 0.5$ and $q_{Ij}(1) = 0.5$ are sent to check node f_j , or in the log domain $\text{LLR}(q_{Ij}) = 0$. Assume all the other LLRs sent to f_j are non-zero, that is $\{\text{LLR}(c_i)\}_{i \in R_{j/I}} \neq 0$. Then looking at (2.3) on page 11, all outgoing messages from check node f_j will be zero (in the log domain), except the message back to variable node χ_I . More concisely:

$$\text{LLR}(r_{ij}) = \begin{cases} 0 & i \neq I \\ \text{non-zero} & i = I \end{cases}$$

This means if a check node only receives one $\text{LLR}(q_{ij}) = 0$ message, then all except one of the associated variable nodes will not receive any constructive information. However since the node that sent the zero message *will* receive feedback from the check node, it is expected that in the next iteration it will send an improved, non-zero message to the check node. If, however, *more than one* received message is zero, then *all outgoing messages* will be zero. This means that the check node will not be providing any useful information to any node in the graph in that iteration, and will continue to do so until the bits are improved by other nodes.

In a more realistic scenario where the $\text{LLR}(c_I)$ is not zero, but instead quite weak it can still very adversely affect a check node. Since a check node calculation is based on the product of incoming messages, if one very small LLR is sent to a check node, all messages from that node will be significantly reduced, except for the return message to the weak variable node. The system can recover if the other associated bits can provide enough reliable information back to the weak node. If there are two or more weak variable nodes associated with the check node, it will be more difficult to recover since all outgoing messages will be reduced. It is thus desirable for better performance to have at most one weak variable node associated to each check node.

It is not possible to know *exactly* which variable nodes will be weak and which ones will be strong, however using the information gained from Chapter 3 it is known which variable nodes *tend* to be more unreliable than others. This can be used to evenly distribute the most *likely* unreliable bits across the graph, in the hopes that each check node will be associated with at most one weak variable node.

Spreading unreliable bits across the graph mitigates their negative effects, and also has another advantage. It also spreads the reliable bits across the graph, attempting to ensure that each unreliable bit will be positively affected by many reliable nodes.

This will help to provide useful information to the unreliable node and hopefully it will arrive at the correct bit value.

4.2.1 Implementation

Permuting the columns of the parity-check matrix to evenly distribute the unreliable bits is difficult and time consuming. The algorithm used to optimize the distribution is discussed below. This document will define an ‘unreliable bit’ as the least statistically reliable bit in a symbol, based on the modulation scheme. This algorithm assumes that a signal mapping scheme similar to the Gray Mapping used in Chapter 3 has been used.

Notation:

- m = number of rows in the parity-check matrix.
- d_c = Average degree of check-nodes.
- M = number of bits per symbol. It is assumed that the least significant bit is the most unreliable. It is deemed that there is one unreliable bit per symbol.
- unreliable column = a column whose index makes it unreliable, and which has a ‘one’ in the row of interest.
- u_j = number of unreliable columns in row j .
- $U = \lceil \frac{d_c}{M} \rceil$, allowable unreliable columns in any given row.³

³U can be less for irregular codes, due to the variation in d_v . This is explained in Section 5.3.

- $c(j) = \begin{cases} 0 & u_j \leq U \\ 1 & u_j > U \end{cases}$ true (1) if row j has more than the allowable number of unreliable columns, else false (0).
- $z(j) = \begin{cases} 0 & u_j \geq U \\ 1 & u_j < U \end{cases}$ true (1) if row j has less than the allowable number of unreliable columns, else false (0).
- $C = \sum_{j=0}^m c(j)$, number of rows that have more than the allowable number of unreliable columns.
- $Z = \sum_{j=0}^m z(j)$, number of rows that have less than the allowable number of unreliable columns.

The algorithm's purpose is to minimize the number of rows that have more unreliable columns than is allowed. C can be viewed as a cost function, since it represents the total columns that do not conform to the desired distribution.

1. For a given row, j , determine if $u(j)$ is greater than, equal to, or less than U .
 - If $u(j) > U$, it means there are more unreliable columns than is allowed. The algorithm will swap unreliable columns for reliable columns until $u(j) = U$, checking all possibilities. However it will determine which unreliable column(s) to switch with which reliable column(s) based on which combination results in the *minimum* value of C . In the case of a tie in the value of C between two possible permutations, the one that *minimizes* Z will be used, since to have C minimized, it is also desirable to have Z minimized.

- If $u(j) < U$, it means that there is less than the allowed number of unreliable columns. It is desirable to permute the matrix such that $u(j) = U$. Possible permutations will be compared in the same manner as above. The switch that is chosen will have the minimum value of C , and in the case of a tie, the minimum value of Z .
- If $u(j) = U$, there is nothing wrong with the row and nothing needs to be done.

2. If $C = 0$ or C is below the threshold desired by the user: Stop.

$C > 0$ or C is above the threshold: increment j .

It was found that the best performance was achieved when j is run sequentially through all possible values $\{0, 1, 2, \dots, m\}$ at least once, after which j is randomly assigned. This ensures that every row is ‘checked’ at least once, and it does not become stuck in a cycle.

This algorithm, though somewhat crude, is effective at evenly distributing the unreliable bits among the check-nodes. It can permute the matrix such that less than 3% of the rows have more than U unreliable columns in a reasonable of time (based on the matrices used in this document). Since this algorithm takes an existing parity-check matrix and permutes it, it cannot be guaranteed that there exists a solution where $C = 0$.

Chapter 5

Simulation Results

This chapter will discuss the performance of both versions of RBCM using different LDPC codes and modulation schemes. It will be shown that RBCM with CNO significantly outperforms traditional BICM in cases where regular codes are used and their check node degree equals the number of bits per symbol, $d_c = M$. The best performance is achieved when one and only one ‘unreliable’ bit is sent to each check node. RBCM can be performed on regular and irregular codes, but as this chapter will show there is little performance gain when it is applied to irregular codes.

All simulations are done using an LLR version of *Belief Propagation Algorithm*, constructed based on “Pearl’s Belief Propagation in Bayesian Networks”, as implemented in “The Art of Error Correcting Coding” by R. H. Morelos-Zaragoza, `llr_pearl.c` [29].¹

All graphed points in this chapter are generated with *at least* 100 code word errors. In all simulations the maximum number of iterations is 500, and each point in a given graph was generated using the same initial conditions. That is the seed for

¹At the time of writing, there were bugs in the publicly available version. I have since corrected the errors for these simulations.

the Channel Adapters as well as for the AWGN generator are the same for each curve in a graph. In all applicable graphs, the solid lines represent Bit Error Rate(BER), while the dashed lines represent Word Error Rate (WER).

5.1 LDPC Codes Used

All the codes used in our simulations are available on Dr. MacKay's website "Encyclopedia of Sparse Graph Codes" [30]. The following will summarize each code: block length, rate, construction (if applicable), author and node degree distributions.

1. Name: $\text{reg}(1008,504)$

- (n,m,k): (1008,504,504)
- Rate: 1/2
- Author: D.J.C. MacKay
- Regular / Irregular: Regular
- Node Degree Distribution:

Node Degree	Number of Nodes	Fraction of Nodes
Variable Nodes		
3	1008	1
Check Nodes		
6	504	1

Table 5.1: Degree Distribution of $\text{reg}(1008,504)$

2. Name: $\text{reg}(1920,1280)$

- (n,m,k) : $(1920,1280,640)$
- Rate: $1/3$
- Author: D.J.C. MacKay
- Regular / Irregular: Regular (slight difference in variable node degrees)
- Node Degree Distribution:

Node Degree	Number of Nodes	Fraction of Nodes
Variable Nodes		
2	640	$1/3$
3	1280	$2/3$
Check Nodes		
4	1280	1

Table 5.2: Degree Distribution of $\text{reg}(1920,1280)$

3. Name: PEG(504,252)

- (n,m,k): (504,252,252)
- Rate: 1/2
- Method of Construction: Progressive Edge Growth (PEG)
- Author: Xiao-Yu Hu, IBM Zurich research labs
- Regular / Irregular: Irregular
- Node Degree Distribution:

Node Degree	Number of Nodes	Fraction of Nodes
Variable Nodes		
2	241	0.478175
3	141	0.279762
4	18	0.035714
5	49	0.097222
7	4	0.007937
14	1	0.001984
15	50	0.099206
Check Nodes		
7	22	0.087302
8	210	0.833333
9	20	0.079365

Table 5.3: Degree Distribution of PEG(504,252)

4. Name: PEG(1008,504)

- (n,m,k): (1008,504,504)
- Rate: 1/2
- Method of Construction: Progressive Edge Growth (PEG)
- Author: Xiao-Yu Hu, IBM Zurich research labs
- Regular / Irregular: Irregular
- Node Degree Distribution:

Node Degree	Number of Nodes	Fraction of Nodes
Variable Nodes		
2	481	0.477183
3	283	0.280754
4	35	0.034722
5	98	0.097222
7	9	0.008929
14	1	0.000992
15	101	0.100198
Check Nodes		
7	5	0.009921
8	493	0.978175
9	6	0.011905

Table 5.4: Degree Distribution of PEG(1008,504)

5.2 Regular Codes

It is advantageous to use RBCM with regular codes, since the d_c of these codes are uniform, and it is possible to evenly distribute the unreliable bits. Furthermore, as

will be shown later, regular codes do not suffer the negative effects of VNO at high E_b/N_0 .

RBCM:CNO works best if one and only one unreliable bit is mapped to each check node. Since an unreliable bit has been defined as the statistically least reliable in a symbol, this can only occur for regular codes when the d_c of the code is equal to the number of bits per symbol, M , or $d_c = M$.

5.2.1 Results for reg(1008,504)

The reg(1008,504) code has a constant d_c equal to six, thus to achieve one unreliable bit per check node it is necessary to use a modulation scheme with a symbol length of six bits. 64-PSK will be used since $M=6$, and as shown in Section 3.4, the least significant bit is considerably less reliable than the other bits.

The CNO optimization algorithm was successfully applied to this code. Prior to optimization, the randomly interleaved code contained 144 out of 504 check nodes, or 28.6%, with more than one unreliable bit associated to them. After optimization it contained only 11 out 504, or 2.2% of the total check nodes. The code could be further optimized if it was run for long, but the results of the current scheme already show a considerable improvement.

Figure 5.1 shows the performance of BICM compared to RBCM:CNO, using reg(1008,504) over 64-PSK with Gray Mapping. It shows that RBCM:CNO can achieve a gain of 0.4 dB to 0.5 dB for $\text{BER (WER)} < 10^{-2}$ ($< 10^{-1}$), and furthermore that this gain *increases* with E_b/N_0 . This significant gain is achieved with no additional complexity or delay to the system, simply a one-time adjustment to the parity-check matrix based on the type of modulation.

Figure 5.1: Probability of Error Curves for BICM and RBCM:CNO Using $\text{reg}(1008,504)$, 64-PSK Gray Mapping

The average number of iterations for BICM as compared to RBCM:CNO are almost identical for the same BER. Figure 5.2 shows the number of iterations for BICM and RBCM:CNO. The graph illustrates that both curves closely match each other, suggesting that for a given BER each scheme requires the same number of iterations to reach it. However, if both were compared for a given E_b/N_0 RBCM:CNO would show improvement since it yields a lower BER for a given E_b/N_0 .

Figure 5.2: Average Number of Iterations vs. Bit Error Rate for BICM and RBCM:CNO Using reg(1008,504), 64-PSK Gray Mapping

Two Unreliable Bits per Check Node

If unreliable bits are evenly distributed such that there are two unreliable bits per check node the results are seriously degraded, showing no improvement in BER or WER compared to BICM. This is to be expected, since if a check node receives more than one very low LLR messages from a variable node, all outgoing message will be degraded thus making it more difficult for the code to arrive at the correct codeword. The performance difference between having exactly two unreliable bits compared to having more than two unreliable bits associated to a check node is not very large. This is in stark contrast to the performance difference between having exactly one unreliable bit and more than one unreliable bit associated to a check node, which results in significant improvement.

Figure 5.3 shows the performance of BICM compared to RBCM:CNO, using $\text{reg}(1008,504)$ over 8-PSK Gray Mapping. In this case, and this case alone, only 33 codeword errors were simulated. The reason for stopped the simulation early is the lack of improvement in performance. It has been included to illustrate that RBCM:CNO works best if only one unreliable bit is mapped to each check node.

5.2.2 Results for $\text{reg}(1920,1280)$

The results of using $\text{reg}(1920,1280)$ are very similar to those of the previous section. They continue to show the trend that if only a single unreliable bit is mapped to each check node, a performance gain can be expected. The $\text{reg}(1920,1280)$ code has a constant $d_c = 4$ but has a slight difference in its variable node degrees, where one third of the nodes have $d_v = 2$ and the remaining two thirds have $d_v = 3$. With irregular codes the optimization algorithm tends to map low-degree variable nodes

Figure 5.3: Probability of Error Curves for BICM and RBCM:CNO Using $\text{reg}(1008,504)$, 8-PSK Gray Mapping

to unreliable bit positions. This did not occur with this ‘semi-regular’ code, of the unreliable bits one third were mapped to nodes with $d_v = 2$ and two thirds to nodes with $d_v = 3$.

Figure 5.4: Probability of Error Curves for BICM and RBCM:CNO Using reg(1920,1280), 16-PSK Gray Mapping

Figure 5.4 shows the performance of BICM compared to RBCM:CNO, over 16-PSK Gray Mapping. It shows that using RBCM:CNO with reg(1920,1280) produces a gain of about 0.2-0.3 dB for BER (WER) $< 10^{-2}$ ($< 10^{-1}$). This is less than the gain observed for reg(1008,504) and is mainly because in 16-PSK the difference in reliabilities is not as great as in 64-PSK.

The difference in probability of error between the most significant and least significant bit is many times smaller in 16-PSK than in 64-PSK. This acts to reduce the improvement caused by the ‘spreading’ of unreliable bits across the graph.

5.3 Irregular Codes

RBCM can be applied to irregular codes, though little to no performance gain is achieved. A discussion of these results is, none the less, very useful in describing the sensitivity of RBCM to variable node degree changes.

As with regular codes, RBCM:CNO will evenly distribute the unreliable bits across the check nodes, preferably a single unreliable bit to each check node. For regular codes this can only occur when $d_c = M$, however irregular codes can achieve one unreliable bit per check node even when the average d_c is *greater* than M . The average d_c means the average number of variable nodes associated to each check node. However in irregular codes the variable nodes are not evenly distributed, since d_v is not constant. Nodes with high d_v are associated to numerous check nodes, while nodes with low d_v are associated to few. This allows the algorithm to still map one unreliable bit to each check node, even though $d_c > M$. The exact value depends entirely on the node degree distribution, and was done empirically.

As described in Chapter 4 both versions of RBCM, Variable Node Optimization (VNO) and Check Node Optimization (CNO), can be applied to irregular codes. VNO also has two possible versions, either high-reliability or low-reliability bits can be mapped to high degree variable nodes. If high-reliability bits are mapped little to no performance gain is achieved, whereas if low-reliability bits are mapped then a significant performance *reduction* occurs.

This phenomenon can be used to explain why CNO does not give any performance gain for irregular codes, even when only one unreliable bit is mapped to each check node. When optimizing the code for CNO, the idea is that no more than U columns (ideally one) in any given row are unreliable. For irregular codes, would result in mapping the high degree nodes to reliable columns and low degree nodes to unreliable columns. This means that unreliable bits will end up being mapped to variable nodes of low degree, which causes the result to resemble that of VNO. This acts to mitigate the gain achieved by CNO and causes little or no overall improvement in performance.

5.3.1 Results for PEG(504,252)

The PEG(504,252) code has an average $d_c = 8$ (see Table 5.3), but using CNO it can be optimized for 64-PSK ($M=6$), with one unreliable bit per check node. This is possible, as mentioned in the previous section, because while on average it has a $d_c = 8$, the algorithm deliberately chooses the nodes of low degree to be unreliable. This means that there are fewer edges in the graph connected to unreliable nodes than the average would suggest. This allows the mapping of one unreliable bit per check node, despite $d_c > M$.

Figure 5.5 shows the BER performance of PEG(504,252), BICM vs RBCM:CNO. This code was optimized in two different ways: firstly, it was optimized with one unreliable bit per check node (RBCM:CNO $U=1$) and secondly, with two unreliable bits per check node (RBCM:CMO $U=2$). The performance when $U=2$ is roughly the same as for when $U=1$, approximately 0.1 dB better than BICM. This is contrary to what occurred for regular codes, where the difference between mapping one and two bits to a check node was quite significant.

Figure 5.5: Probability of Bit Error Curves Using PEG(504,252), 64-PSK Gray Mapping

This lack of performance gain occurs when optimizing an irregular code. The algorithm, in its effort to minimize the number of rows which have more than U unreliable columns, will tend to have low degree columns mapped to unreliable positions. Table 5.5 shows the unreliable column weight distribution after optimization, which is quite different from the pre-optimized distribution given in Table 5.3. While the average weight of all the columns in the parity-check matrix is 4.0, the average weight of the unreliable columns is only 2.5, indicating that lower degree nodes are being mapped to unreliable column positions. This means that the bits that are less protected by modulation are also less protected by coding. This acts to dominate the BER performance and reduce the performance gain of CNO.

Column Weight	Unreliable Columns	
	Number	Fraction
2	52	0.62
3	28	0.33
5	3	0.04
7	1	0.01

Avg. Unreliable Weight = 2.5

Avg. Reliable Weight = 4.3

Avg. Total Weight = 4.0

Table 5.5: PEG(504,252) Unreliable Column Weight Distribution, for 64-PSK

Lower Order Modulation Schemes

When using lower order modulation schemes, with both CNO and VNO, over irregular codes, little to no performance gain is achieved. This is similar to higher order modulation schemes. The reasons vary for each case, but this is to be expected because the difference in reliabilities for lower order schemes is not as large as for higher order schemes, there is more than one unreliable bit per check node, and thirdly

because the unreliable bits tend to be mapped to low degree nodes.

Figure 5.6: Probability of Bit Error Curves Using PEG(504,252), 4-PAM Gray Mapping

Figure 5.6 shows the BER performance of BICM vs. RBCM:VNO under 4-PAM. The graph ‘VNO1’ refers to when highly-reliable bits are mapped to high degree variable nodes, and ‘VNO2’ refers to situation where low-reliability bits are mapped to high degree nodes. The degree distributions after optimization, for VNO1 and VNO2, are shown in Table 5.6. As expected they are very different, VNO1 causes the average unreliable column weight to be low while VNO2 causes it to be high.

The performance of VNO2 is always significantly less than BICM. As E_b/N_0 in-

Column Weight	Unreliable Columns	
	Number	Fraction
2	241	0.96
3	11	0.04
Avg. Unreliable Weight = 2.04		
Avg. Reliable Weight = 5.94		
Avg. Total Weight = 4.0		
(a) VNO1		

Column Weight	Unreliable Columns	
	Number	Fraction
3	130	0.520
4	18	0.071
5	49	0.190
7	4	0.015
14	1	0.004
15	50	0.200
Avg. Unreliable Weight = 5.94		
Avg. Reliable Weight = 2.04		
Avg. Total Weight = 4.0		
(b) VNO2		

Table 5.6: PEG(504,252) Optimized Column Weight Distributions, for 4-PAM

creases the performance of VNO1 and VNO2 appear to intersect and if the simulation were continued it is conjectured that VNO2 would outperform VNO1. This comes back to the tradeoff between initial and iterative performance. VNO1 offers good initial performance and poor iterative performance while VNO2 offers the opposite.

Figure 5.7 shows the BER performance over 8-PSK of both forms of VNO and of CNO (with $U=2$). This figure shows the same trends as above, with VNO1 outperforming VNO2 initially, but later the trend would change. Neither performs well vs randomly-interleaved BICM, especially at higher E_b/N_0 values. The figure also shows that, as expected, CNO performs very close to VNO1 and does not provide any improvement in performance compared to randomly-interleaved BICM.

5.3.2 Results for PEG(1008,504)

This section will deal with the results for PEG(1008,504). The results are similar to those of PEG(504,252). This demonstrates that codeword length is not an important factor in determining the performance gain of RBCM, for irregular codes.

Figure 5.7: Probability of Bit Error Curves Using PEG(504,252), 8-PSK Gray Mapping

Figure 5.8: Probability of Bit Error Curves Using PEG(1008,504), 64-PSK Gray Mapping

Figure 5.8 shows the BER and WER performance over 64-PSK. It shows the performance of CNO, and both versions of VNO but there is no gain compared to BICM.

Column Weight	Unreliable Columns	
	Number	Fraction
2	336	1
Avg. Unreliable Weight = 2.00		
Avg. Reliable Weight = 5.00		
Avg. Total Weight = 4.00		
(a) VNO1		

Column Weight	Unreliable Columns	
	Number	Fraction
3	92	0.274
4	35	0.104
5	98	0.292
7	9	0.027
14	1	0.003
15	101	0.301
Avg. Unreliable Weight = 7.43		
Avg. Reliable Weight = 2.28		
Avg. Total Weight = 4.00		
(b) VNO2		

Table 5.7: PEG(1008,504) Optimized Column Weight Distributions, for 8-PSK

As with PEG(504,252) both versions of VNO cause the parity-check matrix to look quite different. The degree distributions are presented in Table 5.7.

Figure 5.9 shows the BER performance over 8-PSK, using BICM, CNO and both forms of VNO. As expected VNO2 performs poorly, and VNO1 performs ~ 0.1 dB better than BICM, but as E_b/N_0 increases the performance degrades. CNO gives little to no performance gain, which follows the trend set by PEG(504,252).

Figure 5.9: Probability of Bit Error Curves Using PEG(1008,508), 8-PSK Gray Mapping

Chapter 6

Conclusion

In this paper, we investigated ways of using the known difference in protection offered by non-binary modulation schemes to increase the performance of LDPC codes. A detailed analysis was performed to compare the protection offered by different types of modulation. *Reliability Based Coded Modulation* (RBCM) was introduced to improve performance of BICM using LDPC codes and its results were analyzed. It was shown that by using RBCM based on *Check Node Optimization* (CNO), significant performance gain could be achieved with properly chosen regular LDPC codes and modulation schemes. It was also shown that the proposed scheme provides little to no performance gain for irregular codes.

This work acts firstly, as a proof of concept, that information about the statistical reliability of bits in a symbol can be used to improve the performance of LDPC codes using BICM. It also creates, based on theory and simulation results, some guidelines for designing RBCM schemes. These guidelines are as follows:

1. The performance of RBCM:CNO yields improved performance for regular codes when one unreliable bit is mapped to each check node.

2. Performance of RBCM:CNO with regular codes will be better for higher order modulation schemes, where the difference in reliabilities of bits in a symbol is greater.
3. RBCM:CNO does not change the average number of iterations needed to converge much, for a given BER.
4. Codeword length does not have a large effect on RBCM performance gain.
5. RBCM does not work well with irregular LDPC codes.

This work shows the gain that can be achieved when coding and modulation are jointly designed in the context of BICM. With properly designed BICM schemes based on LDPC codes, significant performance gains compared to the original randomly-interleaved BICM can be achieved with no additional complexity or delay in the system.

6.1 Future Work

There are still many unanswered questions as to how known statistical reliability can be used to improve LDPC coded modulation performance. There could be other methods to use this information, and other ways to improve the current methods.

A very interesting and potentially fruitful vein of research is in irregular codes. It may be possible to modify the RBCM:CNO optimization routine to evenly distribute the unreliable bits across the check nodes without creating the negative effects caused by assigning low-reliability bits to low-degree nodes. This could lead to similar performance gains for irregular codes that were observed for regular ones.

Appendix A

Additional Tables

A.1 16-PSK Gray Mapping

16-PSK gray mapping is as follows, from starting from (1,0) and working clockwise.

0000 0001 0011 0010 0110 0111 0101 0100 1100 1101 1111 1110 1010 1011 1001 1000

A.2 64-PSK Gray Mapping

64-PSK gray mapping is as follows, from starting from (1,0) and working clockwise.

000000 000001 000011 000010 000110 000111 000101 000100 001100 001101 001111
001110 001010 001011 001001 001000 011000 011001 011011 011010 011110 011111
011101 011100 010100 010101 010111 010110 010010 010011 010001 010000 110000
110001 110011 110010 110110 110111 110101 110100 111100 111101 111111 111110
111010 111011 111001 111000 101000 101001 101011 101010 101110 101111 101101
101100 100100 100101 100111 100110 100010 100011 100001 100000

Bibliography

- [1] R. Narayanaswami, “Coded modulation with low density parity check codes,” Master’s thesis, Texas A&M University, June 2001.
- [2] G. Ungerboeck, “Channel coding with multilevel/phase signals,” *IEEE Trans. Inform. Theory*, vol. IT-28, no. 1, pp. 55–67, Jan. 1982.
- [3] —, “Trellis-coded modulation with redundant signal sets, parts 1 and 2,” *IEEE Commun. Mag.*, vol. 25, pp. 5–12, Feb. 1987.
- [4] T. Richardson, M. Shokrollahi, and R. Urbanke, “Design of capacity-approaching irregular low-density parity-check codes,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 619 – 637, Feb. 2001.
- [5] C. Sae-Young, G. Forney, Jr., T. Richardson, and R. Urbanke, “On the design of low-density parity-check codes within 0.0045 db of the shannon limit,” *IEEE Commun. Lett.*, vol. 5, pp. 58 – 60, Feb. 2001.
- [6] R. G. Gallager, “Low-density parity-check codes,” *IRE Trans. on Inform. Theory*, vol. 8, pp. 21 – 28, Jan. 1962.
- [7] —, “Low-density parity-check codes,” Cambridge, MA:MIT Press, 1963.

- [8] D. MacKay and R. Neal, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 32, pp. 1645 – 1646, Aug. 1996.
- [9] —, "Near shannon limit performance of low density parity check codes," *Electronics Letters*, vol. 33, pp. 457 – 458, Mar. 1997.
- [10] H. Imai and S. Hirakawa, "A new multilevel coding method using error correcting codes," *IEEE Trans. Inform. Theory*, vol. IT-23, pp. 371–377, May 1977.
- [11] U. Wachsmann, R. F. H. Fischer, and J. B. Huber, "Multilevel codes: Theoretical concepts and practical design rules," *IEEE Trans. Inform. Theory*, vol. 45, pp. 1361–1391, July 1999.
- [12] J. Hou, P. H. Siegel, K. B. Milstein, and H. D. Pfister, "Capacity-approaching bandwidth-efficient coded modulation schemes based on low-density parity-check codes," *IEEE Trans. Inform. Theory*, pp. 2141–2155, Sept. 2003.
- [13] E. Zehavi, "8-PSK trellis codes for a rayleigh channel," *IEEE Trans. Commun.*, vol. 40, pp. 873–884, May 1992.
- [14] G. Caire, G. Taricco, and E. Biglieri, "Bit-interleaved coded modulation," *IEEE Trans. Inform. Theory*, vol. 44, pp. 927–946, May 1998.
- [15] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, D. A. Spielman, and V. Stemann, "Practical loss-resilient codes," in *Proc. 29th Annu. Symp. Theory of Computing*, 1997, pp. 150 – 159.
- [16] M. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Analysis of low density codes and improved designs using irregular graphs," in *Proc. 30th Annu. Symp. Theory of Computing*, 1998, pp. 249 – 258.

- [17] ———, “Improved low-density parity-check codes using irregular graphs,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 585 – 598, Feb. 2001.
- [18] D. MacKay, S. Wilson, and D. M.C, “Comparison of constructions of irregular gallager codes,” *IEEE Trans. Commun.*, vol. 47, pp. 1449 – 1454, Oct. 1999.
- [19] C. Berrou, A. Glavieux, and P. Thitimajshima, “Near shannon limit error-correcting coding and decoding: Turbo-codes.” in *IEEE International Conference on Communications, ICC 93*, vol. 2, May 1993, pp. 1064 – 1070.
- [20] R. Tanner, “A recursive approach to low complexity codes,” *IEEE Trans. Inform. Theory*, vol. 27, pp. 533 – 547, Sept. 1981.
- [21] W. Ryan, “An introduction to low-density parity-check codes,” Apr. 2001, course notes.
- [22] G.D. Forney, Jr., “On iterative decoding and the two-way algorithm,” in *Proc. Int. Symp. on Turbo Codes and Related Topics*, Brest, France, Sept. 1997, pp. 12 – 25.
- [23] D. MacKay, “Good error-correcting codes based on very sparse matrices,” *IEEE Trans. Inform. Theory*, vol. 45, pp. 399 – 431, Mar. 1999.
- [24] D. J. C. MacKay, *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, Oct. 2003.
- [25] W. Ryan, “A low-density parity-check code tutorial part 1- introduction and overview,” Apr. 2002, course notes. [Online]. Available: http://www.sm.luth.se/csee/courses/sms/005/2002/Files/ryan_tut_1.pdf

- [26] H.-G. Zhang, D.-F. Yuan, P.-M. Ma, and X.-M. Yang, “Low-density parity-check code (LDPC) schemes with BICM,” in *International Conference on Communication Technology Proceedings, 2003. ICCT 2003.*, vol. 2, Apr. 2003, pp. 1148 – 1151.
- [27] J. Hou, P. H. Siegel, K. B. Milstein, and H. D. Pfister, “Multilevel coding with low-density parity-check component codes,” in *Global Telecommunication Conference, 2001. GLOBECOM '01. IEEE*, vol. 2, Nov. 2001, pp. 1016–1020.
- [28] T. Richardson and R. Urbanke, “The capacity of low-density parity-check codes under message-passing decoding,” *IEEE Trans. Inform. Theory*, vol. 47, pp. 599 – 618, Feb. 2001.
- [29] R. H. Morelos-Zaragoza, *The Art of Error Correcting Coding*, 1st ed. John Wiley & Sons, 2002. [Online]. Available: <http://www.the-art-of-ecc.com>
- [30] D. MacKay, “Encyclopedia of sparse graph codes,” website. [Online]. Available: <http://www.inference.phy.cam.ac.uk/mackay/codes/data.html>