Low-complexity detection and decoding for LDPC-coded data storage channels

Shaghaghi, Mahdi

2010

Shaghaghi, M. (2010). Low-complexity detection and decoding for LDPC-coded data storage channels. Master's thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/41623>

<https://doi.org/10.32657/10356/41623>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**LOW-COMPLEXITY DETECTION AND DECODING
FOR
LDPC-CODED DATA STORAGE CHANNELS**

**MAHDI SHAGHAGHI
SCHOOL OF ELECTRICAL & ELECTRONIC ENGINEERING
2010**

**Low-Complexity Detection and Decoding
for
LDPC-Coded Data Storage Channels**

Mahdi Shaghaghi

School of Electrical & Electronic Engineering

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

2010

Acknowledgments

First, I would like to express my sincere gratitude to Associate Professor Guan Yong Liang for his supervision during my studies in the Nanyang Technological University. I appreciate his kindness, patience, and his advices which led me throughout the course of my research in NTU.

I would also like to thank Dr. Cai Kui, and Dr. Qin Zhiliang as my co-supervisors in the Data Storage Institute.

Finally, I would like to express my regards to A*STAR agency for providing the funding for my studies in Singapore.

Contents

Acknowledgments	i
Summary	v
List of Figures	vii
List of Tables	ix
1 Introduction	1
1.1 Motivation and Objective	2
1.2 Major Contributions of the Thesis	3
1.3 Organization of the Thesis	4
2 Low-Density Parity-Check (LDPC) Codes	6
2.1 Linear Block Codes	6
2.2 Cyclic Codes	7
2.3 LDPC Codes	8
2.3.1 Representations of LDPC Codes	9
2.4 LDPC Code Design Approaches	12
2.4.1 Gallager Codes	13
2.4.2 Irregular LDPC Codes	14
2.4.3 Finite Geometries Codes	15

CONTENTS**iii**

2.5	The General Structure of Finite Geometry LDPC Codes	16
2.6	Finite Geometry Basics	19
2.6.1	Construction of Galois Field $GF(2^m)$	19
2.6.2	Euclidean Geometry $EG(m, 2^s)$	23
2.7	Euclidean Geometry LDPC Codes	26
2.7.1	Type I EG-LDPC Codes	27
2.7.2	Type II EG-LDPC Codes	29
2.8	Projective Geometry LDPC Codes	31
3	Normalized and Offset Min-Sum Decoding of LDPC Codes	32
3.1	Belief Propagation Algorithm	33
3.1.1	Probability Domain BP Algorithm	35
3.1.2	Log Domain BP Algorithm	37
3.2	Min-Sum Algorithm	41
3.3	Normalized and Offset Min-Sum Algorithm	43
3.3.1	Simulation Results	45
4	Detection and Decoding Over Partial Response Channels	49
4.1	BCJR Channel Detector	51
4.1.1	Probability Domain BCJR Algorithm	51
4.1.2	Log Domain BCJR Algorithm	55
4.2	NOMS Algorithm Over PR Channels	58
4.3	Simulation Results	59
5	Markov Chain Monte Carlo Based Detection Over 2-D Magnetic Recording Channels	61
5.1	System Model for 2-D ISI Channels	62
5.2	Detection methods over 2-D ISI Channels	64

CONTENTS	iv
5.3 Monte Carlo Statistical Methods	66
5.3.1 Monte Carlo Integration	67
5.3.2 Markov Chain Monte Carlo (MCMC) Based Sampling	68
5.4 MCMC Based 2-D Detection Method	71
5.4.1 Probability Domain MCMC Based Detector	72
5.4.2 Log Domain MCMC Based Detector	74
5.4.3 Parallel Sampling	77
5.5 Discussion on Computational Complexity	78
5.6 Simulation Results	80
6 Conclusion and Recommendations	86
6.1 Conclusion	86
6.2 Recommendations	87
List of Publications	89
Bibliography	90

Summary

LDPC codes have shown an excellent performance over various channels for different code lengths and rates, and they are finding their way to become the dominant coding scheme for various applications. As an emerging channel coding technique, LDPC codes are widely considered for digital storage systems. In this thesis, LDPC codes based on finite geometries are selected due to their great performance and structure. We will present the details of the construction and features of these codes.

In this thesis, we investigate various detection and decoding techniques that exhibit different performance and complexity trade-offs. The optimal or near optimal methods usually impose an enormous amount of computations, and therefore, they may not be suitable for practical applications. On the other hand, sub-optimal algorithms may suffer from performance degradation. Our main goal in this thesis is to design low-complexity decoders and detectors that have near optimal performance. We first start by introducing a new LDPC decoder, and later, we consider the detection problem. Particularly, we consider low-complexity detection and decoding for LDPC-coded data storage channels.

We introduce a new algorithm for decoding LDPC codes that has a complexity close to low-complexity algorithms, while its performance approaches the performance of the belief propagation (BP) algorithm. The BP algorithm is a well-known method for decoding LDPC codes with a great performance, but it has a high com-

plexity. On the other hand, there is a method known as the min-sum (MS) algorithm which has a much lower complexity but an inferior performance. Various methods have been introduced in the literature to improve the performance of the MS algorithm. As an example, the normalized MS algorithm improves the performance of the MS decoding by normalizing the soft information. Based on the existing methods, we propose a new algorithm that further improves the performance of the MS algorithm.

Next, we consider the problem of detection over 2-dimensional (2-D) inter-symbol interference (ISI) channels. In contrast to 1-D ISI channels, where optimal detection is applicable using the BCJR and Viterbi algorithms, there does not exist a practical implementation of an optimal detection method for 2-D ISI channels because of the prohibitive computational complexity. In the literature, sub-optimal solutions have been considered, but the computational burden of the detection step is still the bottleneck of the receiver side. We propose an algorithm based on Monte Carlo techniques that substantially reduces the complexity, and this will be the main contribution of the thesis.

List of Figures

2.1	Tanner graph of the Example 2.1.	12
3.1	Plot of the $\phi(x)$ function.	40
3.2	Bit and block error probabilities of the type-I 2D (4095, 3367) EG-LDPC code based on the BP and the MS algorithms over the AWGN channel.	43
3.3	Bit error probabilities of the (1023, 781) EG-LDPC code over the AWGN channel.	46
3.4	Bit error probabilities of the (1023, 781) EG-LDPC code over the uncorrelated Rayleigh fading channel with ideal CSI.	47
3.5	Bit error probabilities of the (1023, 781) EG-LDPC code over the uncorrelated Rayleigh fading channel without CSI.	47
4.1	Turbo decoding scheme for coded PR channels.	50
4.2	Bit and block error probabilities of the (1023, 781) EG-LDPC code over the MEEPR4 channel.	60
5.1	IRCSDFa block diagram.	66
5.2	Splitting the M -by- N field into overlapping horizontal strips.	66
5.3	Block diagram of the MCMC based detector.	77

LIST OF FIGURES

viii

5.4	Bit and block error probabilities of the IRCSDFA and MCMC based algorithm at the first turbo iteration using the Gallager LDPC code. .	83
5.5	Bit error probabilities of the IRCSDFA and MCMC based algorithm for five turbo iterations using the Gallager LDPC code.	84
5.6	Block error probabilities of the IRCSDFA and MCMC based algorithms for five turbo iterations using the Gallager LDPC code. . . .	85
5.7	Bit and block error probabilities of the IRCSDFA and MCMC based algorithms at the 5th turbo iteration using the EG-LDPC code. . . .	85

List of Tables

2.1	A list of type-I 2-D EG-LDPC codes	30
5.1	Computational requirements of the MCMC based algorithm and the IRCS DFA	79
5.2	Computational complexity comparison of the MCMC-2D algorithm and the IRCS DFA-I	84

Chapter 1

Introduction

Besides turbo codes, low-density parity-check (LDPC) codes form another class of capacity-approaching codes. At the first stage of their invention by Gallager in early 1960s [1], LDPC codes did not get much attention due to their high complexity for their time, and remained dormant for about three decades until rediscovered by Mackay and Neal [2]. Long LDPC codes with iterative decoding have been shown to achieve an error performance only a fraction of a decibel away from the Shannon limit. This discovery makes LDPC codes strong competitors with turbo codes for error control in many communication and digital storage systems where high reliability is required. For instance, LDPC codes have emerged as a new coding scheme in many recent standards such as DVB-S2 [3] and IEEE 802.16e [4].

While the technological innovations in the design and development of the magnetic and optical recording media and heads are the key to achieving high capacity magnetic and optical storage systems, the role of coding and signal processing techniques is increasingly becoming crucial in supporting and augmenting these advancements. In recent years, LDPC codes have received significant interest for coding data in the Tera bytes per square inch (TB/in²) magnetic recording systems. We will focus our attention on the detection and decoding of LDPC codes for these

systems.

An LDPC-coded data storage channel can be modeled as a serial concatenated system, where an LDPC encoder precedes a finite impulse response (FIR) filter representing the inter-symbol interference (ISI) caused by the channel memory. First, K user data bits are encoded by an LDPC encoder into a codeword of length N . Next, the recorded sequence follows the bipolar signalling, and passes over the FIR filter of the channel. Finally, noise is considered to be added to the filtered sequence. Then, the received sequence (the signal read back from the storage medium) is processed in order to retrieve the transmitted (recorded) data bits. There are two major signal processing operations at the receiver side. One deals with the ISI imposed by the channel memory, and the other performs decoding of the LDPC code. The ISI might be one or two-dimensional (1-D or 2-D) in different storage channels. We will consider the detection problem for both 1-D and 2-D ISI channels.

1.1 Motivation and Objective

A key feature of LDPC codes is that they can be decoded with reasonable complexity by the so-called *belief propagation* (BP) algorithm, which approximates the bitwise MAP decoding. The algorithm would produce the exact marginal *a posteriori* probabilities if the bipartite graph associated with the parity-check matrix of the code, known as Tanner graph, was free of cycles; this is not, however, the case for most LDPC codes, and it makes the BP algorithm sub-optimal. Specially, the degradation is more considerable when the Tanner graph of the code contains a large number of short cycles. Besides, this algorithm may still be regarded too complex for general applications. For these reasons, there has been an intensive effort in the past decade to improve the performance of the BP algorithm as well as to reduce

its complexity.

One of our targets is to investigate different reduced complexity schemes of decoding LDPC codes, bearing in mind that we aim to design a decoder that offers a better performance and/or has a lower complexity than the existing algorithms.

For the detection problem, it is known that optimal detection over 2-D ISI channels is an NP-complete problem, and therefore, its implementation is impractical. There are a number of sub-optimal detection methods for 2-D ISI channels in the literature that have much lower complexity compared to the optimal detection approach. However, the computational complexity of these methods is still considered to be too high. This problem became our motivation for designing a detector with lower complexity.

In a nutshell, the high complexity of the existing detection and decoding algorithms motivated us to look for approaches with lower complexity. Our objective is to design low-complexity detection and decoding algorithms with a performance close to optimal or near optimal methods.

1.2 Major Contributions of the Thesis

The major contributions of the thesis include:

Normalized and Offset Min-Sum Decoder

We introduced a new decoding algorithm for LDPC codes. This algorithm is named normalized and offset min-sum (NOMS) decoding, as it is a generalized form of the normalized min-sum (NMS) and offset min-sum (OMS) algorithms. The NOMS method has a complexity similar to NMS and OMS algorithms, while it outperforms these methods. We investigated the performance of the NOMS algorithm over different channels including the AWGN, uncorrelated Rayleigh fading, and partial re-

sponse (PR) channels. For the case of PR channels, we considered a one-dimensional (1-D) ISI channel, and we used an optimal detection method, known as the BCJR algorithm, to deal with the ISI. The detector was used in conjunction with our proposed decoding method in an iterative scheme known as turbo processing.

Two-dimensional Markov Chain Monte Carlo Based Detector

We developed a novel technique for detection over two-dimensional (2-D) ISI channels. The proposed algorithm is based on Markov chain Monte Carlo (MCMC) techniques, and we refer to it as the 2-D MCMC based detector. This algorithm has significantly lower computational complexity compared with the existing methods, while its performance is at the same level of near optimal detection algorithms. Furthermore, the complexity of our algorithm grows polynomially with the ISI size, whereas the complexity of the trellis based methods, which have similar performance to our method, grows exponentially with the ISI size. We presented the algorithm in both probability and log domains. Our method is a soft-in soft-out (SISO) algorithm, and therefore, it can be used in the context of the turbo decoding method. We used LDPC codes for the purpose of error control. The detector and the decoder pass the soft information to each other in an iterative manner. We showed how the performance improves by increasing the number of iterations. We presented the simulation results over a 2-D ISI channel whose coefficients correspond to a bit-patterned magnetic recording (BPMR) system.

1.3 Organization of the Thesis

In Chapter 2, we will review LDPC codes and some of the design approaches. Then, we will focus on LDPC codes based on finite geometries. In Chapter 3, we start by reviewing various methods of decoding LDPC codes, and then, we will intro-

duce a new algorithm. We will investigate the performance of the proposed method over additive white Gaussian noise (AWGN) and the uncorrelated Rayleigh fading channels, and this will be the first contribution of the thesis. Next, we will consider the 1-D ISI channel in Chapter 4. We start this chapter with an overview of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm which is the optimal method for channel detection. Then, we investigate the performance of our previously proposed algorithm over the 1-D ISI channel. We will consider detection and decoding over 2-D ISI channels in Chapter 5. In this chapter, we will focus on the detection part without getting much involved in the decoding algorithm, and we will use a conventional LDPC decoder. Then, we will propose a novel channel detection algorithm based on Monte Carlo methods which will significantly reduce the computational complexity. The introduced algorithm will be the main contribution of the thesis. Finally, the conclusion and suggestions for future work come in Chapter 6.

Chapter 2

Low-Density Parity-Check (LDPC) Codes

We start this chapter by reviewing LDPC codes, and we present some of the LDPC code design approaches. Then, we will focus on LDPC codes based on finite geometries.

2.1 Linear Block Codes

In digital communication systems, channel coding is used to lower the probability of making a wrong decision at the receiving end. Channel coding techniques make advantage of the fact that the error probability is a function of the distance between the points in the signal constellation. Assuming fixed signal energy and data rate, the Euclidean distance between constellation points can still be increased by increasing the dimension of the signal space. Assume that a symbol consists of k bits and is sent in a time interval of duration T . The method of channel coding is to increase the Euclidean distance between constellation points by sending symbols of length N ($N > k$) in the same time duration T . In this way, a lower bit error rate (BER)

is achieved in the expense of an increase in the transmitted signal bandwidth. The ratio

$$R_c = k/N \quad (2.1)$$

is called the *code rate* and is equal to the inverse of bandwidth expansion imposed by coding.

A (N, k) block code $\mathbf{C}$ is a k dimensional subspace of an N dimensional space of binary N -tuples, and consists of 2^k codewords; each codeword is a binary sequence of length N [5]. A block code is called linear if any linear combination of two codewords is also a codeword. Therefore, every codeword can be considered as a linear combination of k linearly independent codewords. Then, we can write

$$\mathbf{c} = \mathbf{m}\mathbf{G}, \quad (2.2)$$

where $\mathbf{c}$ is the codeword vector, $\mathbf{G}$ is the vector of the k linearly independent codewords which form a $k \times N$ matrix, known as the *generator matrix*, and $\mathbf{m}$ is the message vector which is a binary sequence of length k . For a linear block code, there exists a *parity-check matrix* $\mathbf{H}$ such that

$$\mathbf{c}\mathbf{H}^t = \mathbf{0}, \quad \forall \mathbf{c} \in \mathbf{C}, \quad (2.3)$$

where the superscript “ t ” represents the transpose operation.

It can be shown that $\mathbf{H}$ is the generator matrix of the dual code $\mathbf{C}^\top$ [5], and it is concluded that

$$\mathbf{G}\mathbf{H}^t = \mathbf{0}. \quad (2.4)$$

2.2 Cyclic Codes

Cyclic codes are a subset of linear block codes for which easily implementable encoders and decoders exist. A *cyclic code* is defined as a linear block code with the

extra condition that if $\mathbf{c} = (c_0, c_1, \dots, c_{N-2}, c_{N-1})$ is a codeword, a cyclic shift of it, $\mathbf{c}^{(1)} = (c_{N-1}, c_0, c_1, \dots, c_{N-3}, c_{N-2})$, is also a codeword. The message and code polynomials are defined as

$$m(X) = m_0 + m_1X + \dots + m_{k-1}X^{k-1}, \text{ and} \quad (2.5)$$

$$c(X) = c_0 + c_1X + \dots + c_{N-1}X^{N-1}. \quad (2.6)$$

In any cyclic code all codeword polynomials are multiples of a polynomial of degree $N - k$ of the form

$$g(X) = 1 + g_1X + \dots + g_{N-k-1}X^{N-k-1} + X^{N-k} \quad (2.7)$$

called the *generator polynomial*, where $g(X)$ divides $X^N + 1$. Therefore the codeword polynomial corresponding to $\mathbf{m}$ is given by

$$c(X) = m(X)g(X). \quad (2.8)$$

2.3 LDPC Codes

This section provides the foundations for the study and practice of LDPC codes. We will start with the fundamental representations of LDPC codes via parity-check matrices and Tanner graphs, and classification of LDPC ensembles via Tanner graph degree distributions will be introduced. Next, we will review some of the LDPC code construction techniques which have appeared in the literature. Our attention will be mainly focused on LDPC codes based on finite geometries which have been shown to exhibit very good BER performances for various code lengths and rates [6]. These codes have the advantage of belonging to the cyclic or quasi-cyclic classes of block codes, and lend themselves to simple encoder implementation via shift register circuits. A quasi-cyclic code is a linear code for which cyclically shifting a codeword

a fixed number of times $n_0 \neq 1$ (or a multiple of n_0) of symbol positions either to the right or to the left results in another codeword.

2.3.1 Representations of LDPC Codes

Matrix Representation

Although LDPC codes can be generalized to non-binary alphabets, we shall consider only binary LDPC codes for the sake of simplicity. As described in Section 2.1, a linear block code $\mathbf{C}$ of length N is uniquely specified by either a generating matrix $\mathbf{G}$ or a parity-check matrix $\mathbf{H}$. If it is specified by a parity-check matrix $\mathbf{H}$, the code $\mathbf{C}$ is simply the null space of $\mathbf{H}$. An N -tuple $\mathbf{c} = (c_0, c_1, \dots, c_{N-2}, c_{N-1})$ is a codeword if and only if $\mathbf{c}\mathbf{H}^t = \mathbf{0}$. This simply says that the code bits of a codeword in $\mathbf{C}$ must satisfy a set of parity-check equations specified by the rows of $\mathbf{H}$. LDPC codes can be specified in terms of their parity-check matrices.

Definition 2.1. An LDPC code is defined as the null space of a parity-check matrix $\mathbf{H}$ that has the following structural properties: 1) each row consists of ρ 1's; 2) each column consists of λ 1's; 3) the number of 1's in common between any two columns, denoted by γ , is no greater than 1; that is, $\gamma = 0$ or 1; and 4) both ρ and λ are small compared with the length of the code and the number of rows in $\mathbf{H}$. $\square$

Properties (1) and (2) simply say that the parity-check matrix $\mathbf{H}$ has constant row and column weights ρ and λ , respectively. Property (3) is equivalent to saying that no two rows of $\mathbf{H}$ have more than one 1 in common. Because both ρ and λ are small compared with the code length and the number of rows in the matrix, $\mathbf{H}$ therefore has a small density of 1's. For this reason, $\mathbf{H}$ is said to be *low-density parity-check* matrix, and the code specified by $\mathbf{H}$ is hence called an LDPC code.

The LDPC code given by Definition 2.1 is called a (λ, ρ) – *regular* LDPC code.

The code rate $R_c = k/N$ is related to these parameters via $R_c = 1 - \lambda/\rho$ (this assumes $\mathbf{H}$ is full rank). Note that the rows of $\mathbf{H}$ are not necessarily linearly independent over $\text{GF}(2)$. In this case, to determine the dimension of the code, it is necessary to determine the rank of $\mathbf{H}$. If all the columns or all the rows of the parity-check matrix $\mathbf{H}$ do not have the same weight, an LDPC code is then said to be *irregular*. It is easiest to see the sense in which an LDPC code is regular or irregular through its graphical representation.

Graphical Representation

Tanner considered LDPC codes (and a generalization) and showed how they may be represented effectively by a so-called bipartite graph. A bipartite graph is a graph (nodes connected by edges) whose nodes may be separated into two types, and edges may only connect two nodes of different types. The two types of nodes in a Tanner graph are the *variable nodes* and the *check nodes*. The Tanner graph of a code is drawn according to the following rule: check node m is connected to variable node n whenever element h_{mn} in $\mathbf{H}$ is a 1. One may deduce from this that there are $J = N - k$ check nodes, one for each check equation, and N variable nodes, one for each code bit c_n . Further, the J rows of $\mathbf{H}$ specify the J check node connections, and the N columns of $\mathbf{H}$ specify the N variable node connections.

Example 2.1. Consider a $(10, 5)$ linear block code with $\lambda = 2$ and $\rho = \lambda(N/J) = 4$ with the following $\mathbf{H}$ matrix:

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

The Tanner graph corresponding to $\mathbf{H}$ is depicted in Fig. 2.1. Observe that variable nodes c_0, c_1, c_2 , and c_3 are connected to check node f_0 in accordance with the fact that, in the zeroth row of $\mathbf{H}$, $h_{00} = h_{01} = h_{02} = h_{03} = 1$ (all others are zero). Observe that analogous situations hold for check nodes f_1, f_2, f_3 , and f_4 which correspond to rows 1, 2, 3, and 4 of $\mathbf{H}$, respectively. Note, as follows from the fact that $\mathbf{cH}^t = \mathbf{0}$, the bit values connected to the same check node must sum to zero. We may also proceed along columns to construct the Tanner graph. For example, note that variable node c_0 is connected to check nodes f_0 and f_1 in accordance with the fact that, in the zeroth column of $\mathbf{H}$, $h_{00} = h_{10} = 1$. $\square$

Note that the Tanner graph in this example is regular: each variable node has two edge connections, and each check node has four edge connections (that is, the *degree* of each variable node is 2, and the degree of each check node is 4). This is in accordance with the fact that $\lambda = 2$ and $\rho = 4$.

For irregular LDPC codes, the parameters λ and ρ are functions of the column and row numbers, and so, such notation is not generally adopted in this case. Instead, it is usual in the literature (following [7]) to specify the variable node and check node *degree distribution polynomials*, denoted by $\lambda(x)$ and $\rho(x)$, respectively. In the polynomial

$$\lambda(x) = \sum_{d=1}^{d_v} \lambda_d x^{d-1}, \quad (2.9)$$

λ_d denotes the fraction of all edges connected to degree- d variable nodes, and d_v denotes the maximum variable node degree. Similarly, in the polynomial

$$\rho(x) = \sum_{d=1}^{d_c} \rho_d x^{d-1}, \quad (2.10)$$

ρ_d denotes the fraction of all edges connected to degree- d check nodes, and d_c denotes the maximum check node degree. Note for the regular code above, for which $\lambda = d_v = 2$ and $\rho = d_c = 4$, we have $\lambda(x) = x$ and $\rho(x) = x^3$.

Figure 2.1: Tanner graph of the Example 2.1.

A *cycle* (or loop) of length ℓ in a Tanner graph is a path comprising ℓ edges which closes back on itself. The Tanner graph in the above example possesses a length-6 cycle as exemplified by the six bold edges in the Fig. 2.1. The *girth* g of a Tanner graph is the minimum cycle length of the graph. The shortest possible cycle in a bipartite graph is clearly a length-4 cycle, and such cycles manifest themselves in the $\mathbf{H}$ matrix as four 1's that lie on the corners of a submatrix of $\mathbf{H}$. We are interested in cycles, particularly short cycles, because they degrade the performance of the iterative decoding algorithm used for LDPC codes. This fact will be made evident in the discussion of the iterative decoding algorithm.

2.4 LDPC Code Design Approaches

LDPC codes are generally constructed based on their parity-check matrices. A large number of design techniques exist in the literature, and we introduce some of the more prominent ones in this section. The design approaches target different design criteria, including efficient encoding and decoding, BER performance for various channel models, minimum distance of the code, girth of the Tanner graph, and low-error rate floors.

Construction of LDPC codes falls into two categories, namely random and structured construction. The randomly constructed LDPC codes (regular or irregular) normally demonstrate a better performance compared to their structured counterparts. However, routing becomes very expensive when implementing random codes in hardware (VLSI), and full parallelization requires too much hardware for long or medium-length codes. On the contrary, structured LDPC codes have the advantage of possessing low-complexity encoder and decoders [8, 9]. These codes are generally in the form of cyclic or quasi-cyclic codes, and hence, they are linear-time encodable using shift registers.

2.4.1 Gallager Codes

The original LDPC codes due to Gallager [1] are regular LDPC codes with an $\mathbf{H}$ matrix of the form

$$\mathbf{H} = \begin{bmatrix} \mathbf{H}_1 \\ \mathbf{H}_2 \\ \vdots \\ \mathbf{H}_\lambda \end{bmatrix} \quad (2.11)$$

where the submatrices $\mathbf{H}_d$ have the following structure. For any integers μ and ρ greater than 1, each submatrix $\mathbf{H}_d$ is $\mu \times \mu\rho$ with row weight ρ and column weight 1. The submatrix $\mathbf{H}_1$ has the following specific form: for $i = 1, 2, \dots, \mu$, the i th row contains all of its ρ 1's in columns $(i-1)\rho+1$ to $i\rho$. The other submatrices are simply column permutations of $\mathbf{H}_1$. It is evident that $\mathbf{H}$ is regular, has dimension $\mu\lambda \times \mu\rho$, and has row and column weights ρ and λ , respectively. The absence of length-4 cycles in $\mathbf{H}$ is not guaranteed, but they can be avoided via computer design of $\mathbf{H}$. Gallager showed that the ensemble of such codes has excellent distance properties provided $\lambda \geq 3$ and $\rho > \lambda$.

2.4.2 Irregular LDPC Codes

Richardson *et al.* [7] defined ensembles of irregular LDPC codes parameterized by the degree distribution polynomials $\lambda(x)$ and $\rho(x)$, and showed how to optimize these polynomials for a variety of channels. Optimality is in the sense that, assuming belief propagation decoding, a typical code in the ensemble is capable of reliable communication in the worst channel condition when compared with codes that are outside the ensemble. The worst-case channel condition is called the *decoding threshold*, and the optimization of $\lambda(x)$ and $\rho(x)$ is found by a combination of a so-called *density evolution* (DE) algorithm and an optimization algorithm. In the DE algorithm, the iterative decoder is fed with the probability density functions (pdf's) of the channel output in the first iteration, and the evolution of these pdf's is predicted for the next iterations. Then, the bit error probability is obtained using these predicted pdf's. Having worked the bit error probability out, the decoding threshold is found by the following definition [10].

Definition 2.2. (Threshold of Degree Distribution Pair). The *threshold* associated with the degree distribution pair (λ, ρ) , call it $\sigma(\lambda, \rho)$, is defined as

$$\sigma(\lambda, \rho) \triangleq \sup\{\sigma : P_{(\lambda, \rho)}^i(\sigma) \xrightarrow{i \rightarrow \infty} 0\},$$

where $P_{(\lambda, \rho)}^i(\sigma)$ is the bit error probability at the i th iteration. $\square$

At the next stage, an optimization method is utilized to derive the optimum degree distribution pair for which the threshold is maximum.

Using this approach, an irregular LDPC code has been designed whose simulated performance is within 0.0045 dB of the capacity limit for a binary-input AWGN channel [11]. This code has length $N = 10^7$ and rate $R_c = 1/2$. It is generally true that designs via density evolution are best applied to codes whose length is not too short ($N \gtrsim 5000$) since the DE design algorithm assumes $N \rightarrow \infty$. Thus, the (λ, ρ)

pairs which are optimal for very long codes, will not be optimal for medium-length and short codes.

2.4.3 Finite Geometries Codes

In [6], regular LDPC codes are designed based on the points and lines of the finite geometries. The resulting LDPC codes fall into the cyclic and quasi-cyclic classes of block codes, and hence, they provide simple encoder implementation using shift registers. The cyclic finite geometry codes tend to have relatively large minimum distances. Also, short LDPC codes designed using these techniques are generally better than short LDPC codes designed using pseudo-random $\mathbf{H}$ matrices. Gallager and irregular LDPC codes lack structure, and their hardware implementation is difficult. On the contrary, finite geometry codes are cyclic or quasi-cyclic, and this makes the encoder and the decoder much simpler in terms of the hardware complexity. Besides, finite geometry codes have shown very good performance for finite length codes, whereas random codes or irregular LDPC codes need to have a very long length to perform well. These advantages encouraged us to more focus on LDPC codes based on finite geometries, and we have done most of our simulations using these types of codes.

Well-known finite geometries are Euclidean and projective geometries over finite fields. Based on these two families of finite geometries, four classes of LDPC codes are constructed. The codes of these four classes consist of type-I and type-II Euclidean geometry LDPC codes and type-I and type-II projective geometry LDPC codes. These codes are either cyclic or quasi-cyclic, and therefore, their encoding can be implemented with shift registers. The codes of these four classes are called finite geometry LDPC codes. Finite geometry LDPC codes have relatively good minimum distances, and their Tanner graphs do not contain cycles of length 4.

We will get into the details of constructing finite geometry LDPC codes in future sections.

2.5 The General Structure of Finite Geometry LDPC Codes

This section presents a simple construction of LDPC codes based on the lines and points of finite geometries. Two types of codes are constructed, and their general structural properties are investigated. Lower bounds on their minimum distances are presented.

Let $\mathbf{G}$ be a finite geometry with N points and J lines which has the following fundamental structural properties: 1) every line consists of ρ points; 2) any two points are connected by one and only one line; 3) every point is intersected by λ lines (i.e., every point lies on λ lines); and 4) two lines are either parallel (i.e., they have no point in common), or they intersect at one and only one point. There are two families of finite geometries which have the above fundamental structural properties, namely, Euclidean and projective geometries over finite fields.

Form a $J \times N$ matrix $\mathbf{H}_{\mathbf{G}}^{(1)} = [h_{i,j}]$ over $\text{GF}(2)$ whose rows and columns correspond to the lines and points of the finite geometry $\mathbf{G}$, respectively, where $h_{i,j} = 1$ if and only if the i th line of $\mathbf{G}$ contains the j th point of $\mathbf{G}$ and $h_{i,j} = 0$, otherwise. A row in $\mathbf{H}_{\mathbf{G}}^{(1)}$ simply displays the points on a specific line of $\mathbf{G}$ and has weight ρ . A column in $\mathbf{H}_{\mathbf{G}}^{(1)}$ simply displays the lines that intersect at a specific point in $\mathbf{G}$ and has weight λ . The rows of $\mathbf{H}_{\mathbf{G}}^{(1)}$ are called the incidence vectors of the lines in $\mathbf{G}$, and the columns are called the intersecting vectors of the points in $\mathbf{G}$. Therefore, $\mathbf{H}_{\mathbf{G}}^{(1)}$ is the incidence matrix of the lines in $\mathbf{G}$ over the points in $\mathbf{G}$. It follows from the second structural property of $\mathbf{G}$ that every two columns have exactly one “1-

component” in common, and it follows from the fourth structural property of $\mathbf{G}$ that any two rows have at most one “1-component” in common. The density of this matrix, denoted r , is defined as the ratio of the total number of “ones” in $\mathbf{H}$ to the total number of entries in $\mathbf{H}$. Then we readily see that $r = \rho/N = \lambda/J$. If ρ and λ are small compared to N and J , then $\mathbf{H}_{\mathbf{G}}^{(1)}$ is a low-density matrix which has all the structural properties of Definition 2.1.

The null space over $\text{GF}(2)$ of $\mathbf{H}_{\mathbf{G}}^{(1)}$ gives a binary LDPC code of length N . Such a code is called the type-I geometry- $\mathbf{G}$ LDPC code, denoted $\mathbf{C}_{\mathbf{G}}^{(1)}$. The rows of $\mathbf{H}_{\mathbf{G}}^{(1)}$ are not necessarily linearly independent. Let $\mathcal{R}$ be the rank of $\mathbf{H}_{\mathbf{G}}^{(1)}$. Then $\mathbf{C}_{\mathbf{G}}^{(1)}$ is a binary $(N, N - \mathcal{R})$ linear code with $\mathbf{H}_{\mathbf{G}}^{(1)}$ as its parity-check matrix.

Let $\mathbf{H}_{\mathbf{G}}^{(2)}$ be the transpose of $\mathbf{H}_{\mathbf{G}}^{(1)}$, i.e., $\mathbf{H}_{\mathbf{G}}^{(2)} = [\mathbf{H}_{\mathbf{G}}^{(1)}]^t$. Then $\mathbf{H}_{\mathbf{G}}^{(2)}$ is also a low-density matrix with row weight λ and column weight ρ . The null space over $\text{GF}(2)$ of $\mathbf{H}_{\mathbf{G}}^{(2)}$ gives a binary LDPC code of length J , denoted $\mathbf{C}_{\mathbf{G}}^{(2)}$. Since $\mathbf{H}_{\mathbf{G}}^{(1)}$ and $\mathbf{H}_{\mathbf{G}}^{(2)}$ have the same rank $\mathcal{R}$, $\mathbf{C}_{\mathbf{G}}^{(2)}$ is a binary $(J, J - \mathcal{R})$ linear code. This code is called the type-II geometry- $\mathbf{G}$ LDPC code. $\mathbf{C}_{\mathbf{G}}^{(1)}$ and $\mathbf{C}_{\mathbf{G}}^{(2)}$ are called companion codes and have the same number of parity-check symbols.

It is shown in [6] that the minimum distance $d_{\mathbf{G}}^{(1)}$ of the type-I geometry- $\mathbf{G}$ LDPC code is at least $\lambda + 1$, and $\mathbf{C}_{\mathbf{G}}^{(2)}$ has a minimum distance $d_{\mathbf{G}}^{(2)}$ at least $\rho + 1$.

Both types of geometry- $\mathbf{G}$ LDPC codes are regular and hence their Tanner graphs are regular. Since the row and column weights of $\mathbf{H}_{\mathbf{G}}^{(1)}$ are ρ and λ , respectively, the degrees of each check node and each variable node in the Tanner graph of the type-I geometry- $\mathbf{G}$ LDPC code $\mathbf{C}_{\mathbf{G}}^{(1)}$ are ρ and λ , respectively. Since $\mathbf{H}_{\mathbf{G}}^{(2)}$ is the transpose of $\mathbf{H}_{\mathbf{G}}^{(1)}$, the degrees of each check node and each variable node in the Tanner graph of the type-II geometry- $\mathbf{G}$ code $\mathbf{C}_{\mathbf{G}}^{(2)}$ are λ and ρ , respectively. In fact, the Tanner graphs of the type-I and type-II geometry- $\mathbf{G}$ LDPC codes are dual graphs, i.e., the variable nodes of one graph become the check nodes of the

other graph and the check nodes of one graph become the variable nodes of the other graph.

It follows from the structural properties of the parity-check matrices $\mathbf{H}_{\mathbf{G}}^{(1)}$ and $\mathbf{H}_{\mathbf{G}}^{(2)}$ that no two code bits are checked simultaneously by two check sums. This implies that the Tanner graphs of both types of geometry- $\mathbf{G}$ LDPC codes do not contain cycles of length 4. However, they do contain cycles of length 6. To show this, we use the fundamental property of a finite geometry that any two points are connected by a line. Let p_1 and p_2 be any two points in the finite geometry $\mathbf{G}$. Then there is a line $\mathcal{L}_1$ connecting p_1 and p_2 . Let p_3 be a third point in $\mathbf{G}$ but not on $\mathcal{L}_1$. Then there is a line $\mathcal{L}_2$ connecting p_1 and p_3 , and a line $\mathcal{L}_3$ connecting p_2 and p_3 . These three lines enclose a triangle with p_1 , p_2 , and p_3 as the vertices. In the Tanner graph of the type-I geometry- $\mathbf{G}$ LDPC code, these three lines $\mathcal{L}_1$, $\mathcal{L}_2$, and $\mathcal{L}_3$ correspond to three check nodes, say f_1 , f_2 , and f_3 , and the three points p_1 , p_2 , and p_3 correspond to three variable nodes, say c_1 , c_2 , and c_3 . Each of these three check nodes, f_1 , f_2 , and f_3 is connected to only two of the three variable nodes c_1 , c_2 , and c_3 . Since no two check nodes are connected to the same pair of variable nodes, the edges connecting f_1 , f_2 , and f_3 to c_1 , c_2 , and c_3 form a cycle of length 6 in the Tanner graph of the type-I geometry- $\mathbf{G}$ code $\mathbf{C}_{\mathbf{G}}^{(1)}$. The number of cycles of length 6 is equal to the number of triangles in $\mathbf{G}$ which can be enumerated and is equal to [6]

$$N_6 = \frac{1}{6}N(N-1)(N-\rho). \quad (2.12)$$

Since the Tanner graphs of type-I and type-II geometry- $\mathbf{G}$ codes are dual, they have the same girth and the same cycle distribution. The above analysis shows that the girth of the Tanner graph of an LDPC code constructed based on the lines and points of a finite geometry is 6.

2.6 Finite Geometry Basics

Before getting into the details of the construction of LDPC codes based on finite geometries, it is required that we make a brief review of the mathematical basics on the structure of finite geometries. This section starts from the preliminary concepts of Galois fields, and evolves to the construction of Euclidean and projective geometries.

2.6.1 Construction of Galois Field $\text{GF}(2^m)$

First we consider computations with polynomials whose coefficients are from the binary field $\text{GF}(2)$. A polynomial $f(X)$ with one *variable* X and with coefficients from $\text{GF}(2)$ is of the following form:

$$f(X) = f_0 + f_1X + f_2X^2 + \cdots + f_nX^n, \quad (2.13)$$

where $f_i = 0$ or 1 for $0 \leq i \leq n$. The *degree* of a polynomial is the largest power of X with a nonzero coefficient. For the polynomial above, if $f_n = 1$, $f(X)$ is a polynomial of degree n ; if $f_n = 0$, $f(X)$ is a polynomial of degree less than n . The degree of $f(X) = f_0$ is zero. In the following we use the phrase “a polynomial over $\text{GF}(2)$ ” to mean “a polynomial with coefficients from $\text{GF}(2)$ ”. Polynomials over $\text{GF}(2)$ can be added (or subtracted), multiplied, and divided in the usual way.

A polynomial $p(X)$ over $\text{GF}(2)$ of degree m is said to be *irreducible* over $\text{GF}(2)$ if $p(X)$ is not divisible by any polynomial over $\text{GF}(2)$ of degree less than m but greater than zero. As a case in point, $X^2 + X + 1$ does not have either “0” or “1” as a root, and so is not divisible by any polynomial of degree 1. Therefore, $X^2 + X + 1$ is an irreducible polynomial of degree 2. The polynomial $X^3 + X + 1$ is an irreducible polynomial of degree 3. It has been proved that, for any $m \geq 1$, there exists an irreducible polynomial of degree m . An important theorem regarding irreducible

2.6 Finite Geometry Basics

20

polynomials over $\text{GF}(2)$ is given below without a proof.

Theorem 2.1. Any irreducible polynomial over $\text{GF}(2)$ of degree m divides $X^{2^m-1} + 1$. $\square$

An irreducible polynomial $p(X)$ of degree m is said to be *primitive* if the smallest positive integer n for which $p(X)$ divides $X^n + 1$ is $n = 2^m - 1$. We may check that $p(X) = X^4 + X + 1$ divides $X^{15} + 1$ but does not divide any $X^n + 1$ for $1 \leq n < 15$. Hence, $X^4 + X + 1$ is a primitive polynomial.

Here, we present a method for constructing the Galois field of 2^m elements ($m > 1$) from the binary field $\text{GF}(2)$. We begin with the two elements 0 and 1, from $\text{GF}(2)$ and a new symbol α . Then we define a multiplication “ $\cdot$ ” to introduce a sequence of powers of α as follows:

$$\begin{aligned} 0 \cdot 0 &= 0, 0 \cdot 1 = 1 \cdot 0 = 0, 1 \cdot 1 = 1, \\ 0 \cdot \alpha &= \alpha \cdot 0 = 0, 1 \cdot \alpha = \alpha \cdot 1 = \alpha, \alpha^2 = \alpha \cdot \alpha, \\ &\vdots \\ \alpha^j &= \alpha \cdot \alpha \cdots \alpha \text{ (} j \text{ times)}, \\ &\vdots \end{aligned} \tag{2.14}$$

Now, we have the following set of elements on which a multiplication operation “ $\cdot$ ” is defined:

$$F = \{0, 1, \alpha, \alpha^2, \dots, \alpha^j, \dots\}. \tag{2.15}$$

The element 1 is sometimes denoted α^0 .

Next we put a condition on the element α , so that the set F contains only 2^m elements and is closed under the multiplication “ $\cdot$ ” defined by (2.14). Let $p(X)$ be a primitive polynomial of degree m over $\text{GF}(2)$. We assume that $p(\alpha) = 0$. Since $p(X)$ divides $X^{2^m-1} + 1$ (Theorem 2.1), we have

$$X^{2^m-1} + 1 = q(X)p(X). \tag{2.16}$$

2.6 Finite Geometry Basics

21

If we replace X by α in (2.16), we obtain

$$\alpha^{2^m-1} + 1 = q(\alpha)p(\alpha) = q(\alpha) \cdot 0 = 0, \quad (2.17)$$

which results in

$$\alpha^{2^m-1} = 1. \quad (2.18)$$

Therefore, under the condition that $p(\alpha) = 0$, the set F becomes finite and contains the following elements:

$$F^* = \{0, 1, \alpha, \alpha^2, \dots, \alpha^{2^m-2}\}. \quad (2.19)$$

The nonzero elements of F^* are closed under the multiplication operation “ $\cdot$ ” defined by (2.14).

Our next step is to define an addition operation “ $+$ ” on F^* . For $0 \leq i < 2^m - 1$, we divide the polynomial X^i by $p(X)$ and obtain the following:

$$X^i = q_i(X)p(X) + a_i(X), \quad (2.20)$$

where $q_i(X)$ and $a_i(X)$ are the quotient and the remainder, respectively. The remainder $a_i(X)$ is a polynomial of degree $m - 1$ or less over $\text{GF}(2)$ and is of the following form:

$$a_i(X) = a_{i0} + a_{i1}X + a_{i2}X^2 + \dots + a_{i,m-1}X^{m-1}. \quad (2.21)$$

Since X and $p(X)$ are relatively prime (i.e., they do not have any common factor except 1), X^i is not divisible by $p(X)$. Therefore, for any $i \geq 0$,

$$a_i(X) \neq 0. \quad (2.22)$$

For $0 \leq i, j < 2^m - 1$, and $i \neq j$, we can also show that

$$a_i(X) \neq a_j(X). \quad (2.23)$$

Suppose that $a_i(X) = a_j(X)$. Then it follows from (2.20) that

$$\begin{aligned} X^i + X^j &= [q_i(X) + q_j(X)]p(X) + a_i(X) + a_j(X) \\ &= [q_i(X) + q_j(X)]p(X). \end{aligned} \quad (2.24)$$

This implies that $p(X)$ divides $X^i + X^j = X^i(1 + X^{j-i})$ (assuming that $j > i$). Since X^i and $p(X)$ are relatively prime, $p(X)$ must divide $X^{j-i} + 1$. However, this is impossible since $j - i < 2^m - 1$ and $p(X)$ is a primitive polynomial of degree m which does not divide $X^n + 1$ for $n < 2^m - 1$. Therefore, our hypothesis that $a_i(X) = a_j(X)$ is invalid. As a result, for $0 \leq i, j < 2^m - 1$ and $i \neq j$, we must have $a_i(X) \neq a_j(X)$. Hence, for $i = 0, 1, 2, \dots, 2^m - 2$, we obtain $2^m - 1$ distinct nonzero polynomials $a_i(X)$ of degree $m - 1$ or less. Now, replacing X by α in (2.20), we obtain the following polynomial expression for α^i :

$$\alpha^i = a_i(\alpha) = a_{i0} + a_{i1}\alpha + a_{i2}\alpha^2 + \dots + a_{i,m-1}\alpha^{m-1}. \quad (2.25)$$

From (2.22), (2.23), and (2.25), we see that the $2^m - 1$ nonzero elements, $\alpha^0, \alpha^1, \dots, \alpha^{2^m-2}$ in F^* , are represented by $2^m - 1$ *distinct nonzero polynomials* of α over $\text{GF}(2)$ with degree $m - 1$ or less. The zero element 0 in F^* may be represented by the *zero polynomial*. As a result, the 2^m elements in F^* are represented by 2^m *distinct polynomials* of α over $\text{GF}(2)$ with degree $m - 1$ or less and are regarded as 2^m distinct elements.

Now, we introduce an addition “+” on F^* as follows. For $0 \leq i, j < 2^m - 1$, we define

$$\begin{aligned} 0 + 0 &= 0, \\ 0 + \alpha^i &= \alpha^i + 0 = \alpha^i, \text{ and} \\ \alpha^i + \alpha^j &= (a_{i0} + a_{i1}\alpha + \dots + a_{i,m-1}\alpha^{m-1}) + (a_{j0} + a_{j1}\alpha + \dots + a_{j,m-1}\alpha^{m-1}) \\ &= (a_{i0} + a_{j0}) + (a_{i1} + a_{j1})\alpha + \dots + (a_{i,m-1} + a_{j,m-1})\alpha^{m-1}, \end{aligned} \quad (2.26)$$

where $a_{i,l} + a_{j,l}$ is carried out in modulo-2 addition. From (2.26), we see that, for $i = j$, $\alpha^i + \alpha^i = 0$, and for $i \neq j$, $(a_{i0} + a_{j0}) + (a_{i1} + a_{j1})\alpha + \dots + (a_{i,m-1} + a_{j,m-1})\alpha^{m-1}$ is nonzero and must be the polynomial expression for some α^k in F^* . Hence, the set F^* is closed under the addition “+” defined by (2.26).

It is shown in [12] that the set F^* is a commutative group under the addition operation “+”, and the nonzero elements of F^* form a commutative group under the multiplication operation “.”. Using the polynomial representation for the elements in F^* , we readily see that the multiplication on F^* is distributive over the addition on F^* . Therefore, the set $F^* = \{0, 1, \alpha, \alpha^2, \dots, \alpha^{2^m-2}\}$ is a Galois field of 2^m elements, $\text{GF}(2^m)$. We notice that the addition and multiplication defined on $F^* = \text{GF}(2^m)$ imply modulo-2 addition and multiplication. Hence, the subset $\{0, 1\}$ forms a subfield of $\text{GF}(2^m)$.

In a finite field $\text{GF}(q)$ with q elements, a nonzero element a is said to be *primitive* if the order of a (the *smallest positive integer* n such that $a^n = 1$) is $q-1$. The powers of a primitive element generate all the nonzero elements of $\text{GF}(q)$ [12]. Therefore, α is a primitive element of $\text{GF}(2^m)$.

In our process of constructing $\text{GF}(2^m)$ from $\text{GF}(2)$, we have developed two representations for the nonzero elements of $\text{GF}(2^m)$: the power representation and the polynomial representation. The power representation is convenient for multiplication and the polynomial representation is convenient for addition.

2.6.2 Euclidean Geometry $\text{EG}(m, 2^s)$

Consider all the m -tuples $(a_0, a_1, \dots, a_{m-1})$ with components a_i 's from the Galois field $\text{GF}(2^s)$. There are $(2^s)^m = 2^{ms}$ such m -tuples. These 2^{ms} m -tuples form a vector space over $\text{GF}(2^s)$. The vector addition and scalar multiplication are defined

in the usual way:

$$\begin{aligned}(a_0, a_1, \dots, a_{m-1}) + (b_0, b_1, \dots, b_{m-1}) &= (a_0 + b_0, a_1 + b_1, \dots, a_{m-1} + b_{m-1}), \\ \beta \cdot (a_0, a_1, \dots, a_{m-1}) &= (\beta \cdot a_0, \beta \cdot a_1, \dots, \beta \cdot a_{m-1}),\end{aligned}\quad (2.27)$$

where addition $a_i + b_i$ and multiplication $\beta \cdot a_i$ are carried out in $\text{GF}(2^s)$. In combinatorial mathematics, the 2^{ms} m -tuples over $\text{GF}(2^s)$ are also known to form an m -dimensional *Euclidean geometry* over $\text{GF}(2^s)$, denoted by $\text{EG}(m, 2^s)$. Each m -tuple $\mathbf{a} = (a_0, a_1, \dots, a_{m-1})$ is called a *point* in $\text{EG}(m, 2^s)$. The all-zero m -tuple, $\mathbf{0} = (0, 0, \dots, 0)$, is called the *origin* of the geometry $\text{EG}(m, 2^s)$.

Let $\mathbf{a}$ be a non-origin point in $\text{EG}(m, 2^s)$ (i.e., $\mathbf{a} \neq \mathbf{0}$). Then the 2^s points, $\{\beta\mathbf{a} : \beta \in \text{GF}(2^s)\}$ constitute a *line* (or *1-flat*) in $\text{EG}(m, 2^s)$. For convenience, we use the notation $\{\beta\mathbf{a}\}$ to represent this line. Because this line contains the origin (with $\beta = 0$), we say that $\{\beta\mathbf{a}\}$ passes through the origin. Let $\mathbf{a}_0$ and $\mathbf{a}$ be two linearly independent points in $\text{EG}(m, 2^s)$ (i.e., $\beta_0\mathbf{a}_0 + \beta\mathbf{a} \neq \mathbf{0}$ unless $\beta_0 = \beta = 0$). Then the collection of the following 2^s points,

$$\{\mathbf{a}_0 + \beta\mathbf{a}\}, \quad (2.28)$$

with $\beta \in \text{GF}(2^s)$, constitutes a line in $\text{EG}(m, 2^s)$ that passes through the point $\mathbf{a}_0$. Line $\{\beta\mathbf{a}\}$ and the line $\{\mathbf{a}_0 + \beta\mathbf{a}\}$ do not have any point in common. Suppose that they have a common point. Then, for some β' and β'' in $\text{GF}(2^s)$,

$$\beta'\mathbf{a} = \mathbf{a}_0 + \beta''\mathbf{a}.$$

As a result, $\mathbf{a}_0 + (\beta'' - \beta')\mathbf{a} = \mathbf{0}$. This implies that $\mathbf{a}_0$ and $\mathbf{a}$ are linearly dependent, which is a contradiction to our assumption that $\mathbf{a}_0$ and $\mathbf{a}$ are two linearly independent points in $\text{EG}(m, 2^s)$. Therefore, $\{\beta\mathbf{a}\}$ and $\{\mathbf{a}_0 + \beta\mathbf{a}\}$ do not have any common points. We say that $\{\beta\mathbf{a}\}$ and $\{\mathbf{a}_0 + \beta\mathbf{a}\}$ are *parallel* lines. Note that $\{\beta\mathbf{a}\}$ is simply a one-dimensional subspace of the vector space of all the 2^{ms} m -tuples over $\text{GF}(2^s)$,

and $\{\mathbf{a}_0 + \beta\mathbf{a}\}$ is a *coset* of $\{\beta\mathbf{a}\}$. Let $\mathbf{b}_0$ be a point not on line $\{\beta\mathbf{a}\}$ or on line $\{\mathbf{a}_0 + \beta\mathbf{a}\}$. The line $\{\mathbf{b}_0 + \beta\mathbf{a}\}$ passes through the point $\mathbf{b}_0$ and is parallel to both $\{\beta\mathbf{a}\}$ and $\{\mathbf{a}_0 + \beta\mathbf{a}\}$. In $\text{EG}(m, 2^s)$, for every line passing through the origin, there are $2^{(m-1)s} - 1$ lines parallel to it. A line $\{\beta\mathbf{a}\}$ and the $2^{(m-1)s} - 1$ lines parallel to it are said to form a *parallel bundle*. The $2^{(m-1)s}$ lines in a parallel bundle simply correspond to a one-dimensional subspace of the vector space of all the m -tuples over $\text{GF}(2)$ and its $2^{(m-1)s} - 1$ cosets.

Let $\mathbf{a}_1$ and $\mathbf{a}_2$ be two linearly independent points in $\text{EG}(m, 2^s)$. The lines $\{\mathbf{a}_0 + \beta\mathbf{a}_1\}$ and $\{\mathbf{a}_0 + \beta\mathbf{a}_2\}$ have only one point, $\mathbf{a}_0$, in common. Suppose that they have another point besides $\mathbf{a}_0$ in common. Then, for some $\beta' \neq 0$ and $\beta'' \neq 0$, we have

$$\mathbf{a}_0 + \beta'\mathbf{a}_1 = \mathbf{a}_0 + \beta''\mathbf{a}_2.$$

This equality implies that $\beta'\mathbf{a}_1 - \beta''\mathbf{a}_2 = 0$ and that $\mathbf{a}_1$ and $\mathbf{a}_2$ are linearly dependent. This is a contradiction to the hypothesis that $\mathbf{a}_1$ and $\mathbf{a}_2$ are linearly independent points in $\text{EG}(m, 2^s)$. Therefore, $\{\mathbf{a}_0 + \beta\mathbf{a}_1\}$ and $\{\mathbf{a}_0 + \beta\mathbf{a}_2\}$ have only one point in common, and they both pass through the point $\mathbf{a}_0$. We say that $\{\mathbf{a}_0 + \beta\mathbf{a}_1\}$ and $\{\mathbf{a}_0 + \beta\mathbf{a}_2\}$ intersect at the point $\mathbf{a}_0$. Given a point $\mathbf{a}_0$ in $\text{EG}(m, 2^s)$, there are

$$\frac{2^{ms} - 1}{2^s - 1} \quad (2.29)$$

lines in $\text{EG}(m, 2^s)$ that intersect at $\mathbf{a}_0$ (including the line $\{\beta\mathbf{a}_0\}$ that passes through the origin). This is an important property that will be used to form parity-check sums for the codes presented in the next section. Another important structural property of lines is that any two points are connected by a line. Let $\mathbf{a}_1$ and $\mathbf{a}_2$ be two points in $\text{EG}(m, 2^s)$. Suppose that $\mathbf{a}_1$ and $\mathbf{a}_2$ are linearly dependent. Then, $\mathbf{a}_2 = \beta_i\mathbf{a}_1$ for some element β_i in $\text{GF}(2^s)$. In this case $\mathbf{a}_1$ and $\mathbf{a}_2$ are connected by the line $\{\beta\mathbf{a}_1\}$. Suppose that $\mathbf{a}_1$ and $\mathbf{a}_2$ are linearly independent. Let $\mathbf{a}_3 = \mathbf{a}_1 + \mathbf{a}_2$. Then, $\mathbf{a}_2 = \mathbf{a}_1 + \mathbf{a}_3$, and $\mathbf{a}_1$ and $\mathbf{a}_2$ are connected by the line $\{\mathbf{a}_1 + \beta\mathbf{a}_3\}$. The total

number of lines in $\text{EG}(m, 2^s)$ is

$$\frac{2^{(m-1)s}(2^{ms} - 1)}{2^s - 1}. \quad (2.30)$$

Next, we show that the elements in the Galois field $\text{GF}(2^{ms})$ actually form an m -dimensional Euclidean geometry $\text{EG}(m, 2^s)$. Let α be a primitive element of $\text{GF}(2^{ms})$. Then, the 2^{ms} elements in $\text{GF}(2^{ms})$ can be expressed as powers of α as follows: $\alpha^\infty = 0, \alpha^0 = 1, \alpha^1, \alpha^2, \dots, \alpha^{2^{ms}-2}$. It is known that $\text{GF}(2^{ms})$ contains $\text{GF}(2^s)$ as a subfield. Every element α^i in $\text{GF}(2^{ms})$ can be expressed as

$$\alpha^i = a_{i0} + a_{i1}\alpha + a_{i2}\alpha^2 + \dots + a_{i,m-1}\alpha^{m-1}, \quad (2.31)$$

where $a_{ij} \in \text{GF}(2^s)$ for $0 \leq j < m$. There is a *one-to-one* correspondence between the element α^i and the m -tuple $(a_{i,0}, a_{i,1}, \dots, a_{i,m-1})$ over $\text{GF}(2^s)$. Therefore, the 2^{ms} elements in $\text{GF}(2^{ms})$ may be regarded as the 2^{ms} points in $\text{EG}(m, 2^s)$, and $\text{GF}(2^{ms})$ as the geometry $\text{EG}(m, 2^s)$. In this case, a line passing through the point α^{ℓ_0} consists of the following 2^s points:

$$\alpha^{\ell_0} + \beta\alpha^{\ell_1}, \quad (2.32)$$

where α^{ℓ_0} and α^{ℓ_1} are linearly independent elements in $\text{GF}(2^{ms})$ and $\beta \in \text{GF}(2^s)$.

The field $\text{GF}(2^{ms})$ may be regarded either as an extension field of $\text{GF}(2^s)$ or as an extension field of $\text{GF}(2^m)$. Therefore, $\text{GF}(2^{ms})$ may be regarded either as the m -dimensional Euclidean geometry $\text{EG}(m, 2^s)$ over $\text{GF}(2^s)$ or as the s -dimensional Euclidean geometry $\text{EG}(s, 2^m)$ over $\text{GF}(2^m)$.

2.7 Euclidean Geometry LDPC Codes

Consider the m -dimensional Euclidean geometry over $\text{GF}(2^s)$, $\text{EG}(m, 2^s)$, whose structure was described in the previous section. This geometry consists of 2^{ms} points,

and each point is represented by an m -tuple over $\text{GF}(2^s)$. The point represented by the all-zero m -tuple, $\mathbf{0} = (0, 0, \dots, 0)$, is called the origin. There are

$$\frac{2^{(m-1)s}(2^{ms} - 1)}{2^s - 1} \quad (2.33)$$

lines in $\text{EG}(m, 2^s)$, and each line consists of 2^s points. Each point in $\text{EG}(m, 2^s)$ is intersected by

$$\frac{2^{ms} - 1}{2^s - 1} \quad (2.34)$$

lines. Two lines in $\text{EG}(m, 2^s)$ are either disjoint or they intersect at one and only point. Each line in $\text{EG}(m, 2^s)$ has $2^{(m-1)s} - 1$ lines parallel to it. From (2.33) we can readily see that there are $(2^{ms} - 1)/(2^s - 1)$ bundles of parallel lines, and each parallel bundle consists of $2^{(m-1)s}$ parallel lines.

Let $\text{GF}(2^{ms})$ be the extension field of $\text{GF}(2^s)$. Each element of $\text{GF}(2^{ms})$ can be represented as an m -tuple over $\text{GF}(2^s)$. As pointed out in the previous section, $\text{GF}(2^{ms})$ forms the m -dimensional Euclidean geometry $\text{EG}(m, 2^s)$ over $\text{GF}(2^s)$. The elements of $\text{GF}(2^{ms})$ give the 2^{ms} points of $\text{EG}(m, 2^s)$, and the zero element 0 of $\text{GF}(2^{ms})$ is the origin of $\text{EG}(m, 2^s)$. Let α be a primitive element of $\text{GF}(2^{ms})$. Then, $\alpha^0 = 1, \alpha, \alpha^2, \dots, \alpha^{2^{ms}-2}$ are all the $2^{ms} - 1$ nonorigin points of $\text{EG}(m, 2^s)$.

2.7.1 Type I EG-LDPC Codes

To construct the type-I EG-LDPC code based on $\text{EG}(m, 2^s)$, we form the parity-check matrix $\mathbf{H}_{EG}^{(1)}(m, s)$ over $\text{GF}(2)$ whose rows are the incidence vectors of all the lines in $\text{EG}(m, 2^s)$ that do not pass through the origin and whose columns correspond to the $2^{ms} - 1$ nonorigin points in $\text{EG}(m, 2^s)$. The columns are arranged in the order of $\alpha^0, \alpha^1, \alpha^2, \dots, \alpha^{2^{ms}-2}$, i.e., the i th column corresponds to the point α^i . Then, $\mathbf{H}_{EG}^{(1)}(m, s)$ consists of $N = 2^{ms} - 1$ columns and

$$J = (2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1) \quad (2.35)$$

rows. Because each line in $\text{EG}(m, 2^s)$ consists of 2^s points, each row of $\mathbf{H}_{EG}^{(1)}(m, s)$ has weight $\rho = 2^s$. Since each point in $\text{EG}(m, 2^s)$ is intersected by $(2^{ms} - 1)/(2^s - 1)$ lines, and considering the omission of the lines passing through the origin, each column of $\mathbf{H}_{EG}^{(1)}(m, s)$ has weight $\lambda = (2^{ms} - 1)/(2^s - 1) - 1$. The density r of $\mathbf{H}_{EG}^{(1)}(m, s)$ is

$$r = \frac{\rho}{N} = \frac{2^s}{2^{ms} - 1} \quad (2.36)$$

which is small for $m \geq 2$ and $s \geq 2$. Therefore, $\mathbf{H}_{EG}^{(1)}(m, s)$ is a low-density parity-check matrix.

Let $\mathbf{C}_{EG}^{(1)}(m, s)$ be the null space of $\mathbf{H}_{EG}^{(1)}(m, s)$. Then $\mathbf{C}_{EG}^{(1)}(m, s)$ is a regular LDPC code of length $N = 2^{ms} - 1$. This code is called the type-I m -dimensional EG-LDPC code. Since the column weight of $\mathbf{H}_{EG}^{(1)}(m, s)$ is $\lambda = (2^{ms} - 1)/(2^s - 1) - 1$ the minimum distance of $\mathbf{C}_{EG}^{(1)}(m, s)$ is at least $(2^{ms} - 1)/(2^s - 1)$. Furthermore, it is shown in [6] that $\mathbf{C}_{EG}^{(1)}(m, s)$ is a cyclic code.

It follows from the analysis of finite-geometry LDPC codes given in Section 2.5 that the Tanner graph of the type-I m -dimensional EG code does not contain cycles of length 4. With some modifications to (2.12) (due to the exclusion of the origin), we find that the number of cycles of length 6 is

$$N_{6,EG}^{(1)}(m, s) = \frac{1}{6} \cdot 2^s \cdot (2^{(m-1)s} - 1) \cdot (2^{ms} - 3 \cdot 2^s + 3)(2^{ms} - 1). \quad (2.37)$$

A special subclass of EG-LDPC codes is the class of type-I two-dimensional (2-D) EG-LDPC codes ($m = 2$). For any positive integer $s \geq 2$, the type-I 2-D EG-LDPC code has the following parameters [6]:

Length	$N = 2^{2s} - 1$	
Number of parity bits	$N - k = 3^s - 1$	
Minimum distance	$d_{EG}^{(1)}(2, s) = 2^s + 1$	
Row weight	$\rho = 2^s$	
Column weight	$\lambda = 2^s$.	(2.38)

For this special case, the geometry $EG(2, 2^s)$ contains $2^{2s} - 1$ lines that do not pass through the origin. Therefore, the parity-check matrix $\mathbf{H}_{EG}^{(1)}(2, s)$ of the type-I 2-D EG-LDPC code is a $(2^{2s} - 1) \times (2^{2s} - 1)$ square matrix. Actually, $\mathbf{H}_{EG}^{(1)}(2, s)$ can be constructed easily by taking the incidence vector of a line in $EG(2, 2^s)$ that does not pass through the origin and then cyclically shifting this incidence vector $2^{2s} - 2$ times. This results in $2^{2s} - 1$ incidence vectors for the $2^{2s} - 1$ distinct lines in $EG(2, 2^s)$ that do not pass through the origin. The initial incidence vector and its $2^{2s} - 2$ cyclic shifts form the rows of the parity-check matrix $\mathbf{H}_{EG}^{(1)}(2, s)$. Therefore, $\mathbf{H}_{EG}^{(1)}(2, s)$ is a square circulant matrix. A list of type-I 2-D EG-LDPC codes is given in Table 2.1 [6].

2.7.2 Type II EG-LDPC Codes

Let

$$\mathbf{H}_{EG}^{(2)}(m, s) = [\mathbf{H}_{EG}^{(1)}(m, s)]^t. \quad (2.39)$$

Then $\mathbf{H}_{EG}^{(2)}(m, s)$ is a matrix with $2^{ms} - 1$ rows and

$$J = (2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1) \quad (2.40)$$

columns. The rows of this matrix correspond to the nonorigin points of $EG(m, 2^s)$ and the columns correspond to the lines in $EG(m, 2^s)$ that do not pass through

Table 2.1: A list of type-I 2-D EG-LDPC codes

s	n	k	d_{min}	ρ	λ
2	15	7	5	4	4
3	63	37	9	8	8
4	255	175	17	16	16
5	1023	781	33	32	32
6	4095	3367	65	64	64
7	16383	14197	129	128	128

the origin. Its column and row weights are $\lambda = 2^s$ and $\rho = (2^{ms} - 1)/(2^s - 1) - 1$, respectively. Any two rows of this matrix have exactly one “1-component” in common, and any two columns have at most one “1-component” in common.

The null space of $\mathbf{H}_{EG}^{(2)}(m, s)$ gives an LDPC code of length J . This code is called the type-II m -dimensional EG-LDPC code, and has minimum distance $d_{EG}^{(2)}(m, s)$ at least $2^s + 1$. Since $\mathbf{H}_{EG}^{(1)}(m, s)$ and $\mathbf{H}_{EG}^{(2)}(m, s)$ have the same rank, $\mathbf{C}_{EG}^{(1)}(m, s)$ and $\mathbf{C}_{EG}^{(2)}(m, s)$ have the same number of parity-check symbols. Since the Tanner graphs of $\mathbf{C}_{EG}^{(1)}(m, s)$ and $\mathbf{C}_{EG}^{(2)}(m, s)$ are dual, they have the same cycle distribution. For $m = 2$, since the parity-check matrix $\mathbf{H}_{EG}^{(1)}(2, s)$ of the type-I 2-D EG-LDPC code $\mathbf{C}_{EG}^{(1)}(2, s)$ is a square matrix whose rows are the cyclic shifts of the first row, the rows of $\mathbf{H}_{EG}^{(2)}(2, s)$ are simply permutations of the rows of $\mathbf{H}_{EG}^{(1)}(2, s)$. Therefore, $\mathbf{C}_{EG}^{(1)}(2, s)$ and $\mathbf{C}_{EG}^{(2)}(2, s)$ are identical.

In general, for $m \neq 2$, $\mathbf{C}_{EG}^{(2)}(m, s)$ is not cyclic but it can be put in quasi-cyclic form. To see this, consider the

$$(2^{(m-1)s} - 1)(2^{ms} - 1)/(2^s - 1)$$

lines in $EG(m, 2^s)$ that do not pass through the origin. The incidence vectors of

these lines can be partitioned into

$$K = (2^{(m-1)s} - 1)/(2^s - 1) \quad (2.41)$$

cyclic classes. Each of these K cyclic classes contains $2^{ms} - 1$ incidence vectors of lines which are obtained by cyclically shifting any incidence vector in the class $2^{ms} - 1$ times. For each cyclic class of incidence vectors of lines, we can choose a representative and the rest of the incidence vectors are generated by cyclically shifting this representative. Now we construct a $(2^{ms} - 1) \times K$ matrix $\mathbf{H}_0$, whose K columns are the K representative incidence vectors of the cyclic classes. For $1 \leq i \leq 2^{ms} - 2$, let $\mathbf{H}_i$ be a $(2^{ms} - 1) \times K$ matrix whose columns are the i th (downward) cyclic shifts of the columns of $\mathbf{H}_0$. Form the following matrix:

$$\mathbf{H}_{EG}^{(2)}(m, s) = [\mathbf{H}_0, \mathbf{H}_1, \dots, \mathbf{H}_{2^{ms}-2}]. \quad (2.42)$$

Then the null space of $\mathbf{H}_{EG}^{(2)}(m, s)$ gives a quasi-cyclic type-II m -dimensional EG-LDPC code $\mathbf{C}_{EG}^{(2)}(m, s)$. Every K cyclic shifts of a codeword in $\mathbf{C}_{EG}^{(2)}(m, s)$ is also a codeword in $\mathbf{C}_{EG}^{(2)}(m, s)$. Encoding of quasi-cyclic codes can also be achieved with linear feedback shift registers [8].

2.8 Projective Geometry LDPC Codes

The construction of PG-LDPC codes is based on the lines and points of the projective geometries (PGs) over finite fields. With PG-LDPC codes, we have more choices for the length and the rate of the codes. The steps involved in the code construction and the properties of the resulted codes based on projective geometry are very similar to those for codes constructed based on Euclidean geometry. Hence, we skip the details of the PG-LDPC codes in this thesis, and we refer the interested readers to [6, 12] for further information. In this thesis, we will use EG-LDPC codes for our simulations.

Chapter 3

Normalized and Offset Min-Sum

Decoding of LDPC Codes

An LDPC code can be decoded in various ways, namely: majority-logic (MLG) decoding, bit-flipping (BF) decoding, weighted BF decoding, *a posteriori* probability (APP) decoding, and iterative decoding based on belief propagation (BP) (commonly known as *sum-product algorithm* (SPA)). The first two types are hard-decision decoding, the last two are soft-decision decoding, and the third one is in between. MLG decoding is the simplest one in decoding complexity. BF decoding requires a little more decoding complexity but gives better error performance than the MLG decoding. APP decoding and the SPA decoding provide much better error performance but require much larger decoding complexity than the MLG and BF decodings. The weighted BF offers a good trade-off between error performance and decoding complexity. SPA decoding gives the best error performance among sub-optimal algorithms for decoding LDPC codes, and yet it is practically implementable. APP decoding provides the best error performance; however, it is computationally intractable. There is a simplified version of APP that requires less decoding complexity at the expense of some performance degradation.

In the sequel, we focus our attention to the details of the soft-decision decoders based on the BP algorithm.

3.1 Belief Propagation Algorithm

The BP algorithm iteratively computes the distributions of variables in graph-based models [13–15]. Much like optimal (maximum *a posteriori* probability (MAP)) symbol-by-symbol decoding, we are interested in computing the APP that a given bit in the transmitted codeword $\mathbf{c} = (c_0, c_1, \dots, c_{N-1})$ equals 1, given the received word $\mathbf{y} = (y_0, y_1, \dots, y_{N-1})$. Without loss of generality, let us focus on the decoding of bit c_n so that we are interested in computing the APP

$$Pr(c_n = 1|\mathbf{y})$$

or the APP ratio (also called the likelihood ratio(LR))

$$l(c_n) \triangleq \frac{Pr(c_n = 0|\mathbf{y})}{Pr(c_n = 1|\mathbf{y})}.$$

Later we will extend this to the more numerically stable computation of the log-APP ratio, also called the log-likelihood ratio (LLR):

$$L(c_n) \triangleq \log \left(\frac{Pr(c_n = 0|\mathbf{y})}{Pr(c_n = 1|\mathbf{y})} \right),$$

where here and in the sequel, the natural logarithm is assumed.

The BP algorithm for the computation of $Pr(c_n = 1|\mathbf{y})$, $l(c_n)$, or $L(c_n)$ is an iterative algorithm which is based on the code's Tanner graph. Specifically, we imagine that the variable nodes represent processors of one type, check nodes represent processors of another type, and the edges represent message paths. In one half iteration, each variable node receives some messages from its neighboring check nodes, and then, processes its input messages and passes the resulting

3.1 Belief Propagation Algorithm

34

output messages back to the neighboring check nodes (two nodes are said to be *neighbors* if they are connected by an edge). The output messages passed concern $Pr(c_n = b | \text{input messages})$, $b \in \{0, 1\}$, the ratio of such probabilities, or the logarithm of the ratio of such probabilities.

In the other half iteration, each check node processes its input messages (which are the output messages of the variable nodes from the previous step), and then, passes its resulting output messages back to its neighboring variable nodes. The messages passed concern $Pr(\text{check equation } f_m \text{ is satisfied} | \text{input messages})$, $b \in \{0, 1\}$, the ratio of such probabilities, or the logarithm of the ratio of such probabilities.

After a prescribed maximum number of iterations or after some stopping criterion has been met, the decoder computes the APP, the LR, or the LLR from which decisions on the bits c_n are made. One example stopping criterion is to stop iterating when $\hat{\mathbf{c}}\mathbf{H}^t = \mathbf{0}$, where $\hat{\mathbf{c}}$ is a tentatively decoded codeword.

The BP algorithm assumes that the messages passed are statistically independent throughout the decoding process. When y_n 's are independent, this independence assumption would hold true if the Tanner graph possessed no cycles. Further, the BP algorithm would yield exact APP's (or LR's or LLR's, depending on the version of the algorithm) in this case [13]. However, for a graph of girth g , the independence assumption is only true up to the $(g/2)$ th iteration, after which messages start to loop back on themselves in the graph's various cycles. Still, simulations have shown that the BP algorithm is generally very effective provided length-four cycles are avoided.

In the remainder of this section, we present the “probability domain” version of the BP algorithm (which computes APP's) and its log domain version (which computes LLR's). Our treatment considers the transmission over the AWGN and the uncorrelated Rayleigh fading channels.

3.1.1 Probability Domain BP Algorithm

We start by introducing the following notation:

- $\mathcal{N}(m) = \{\text{variable nodes connected to check node } f_m\},$
- $\mathcal{N}(m) \setminus n =$
 $\{\text{variable nodes connected to check node } f_m \text{ excluding variable node } c_n\},$
- $\mathcal{M}(n) = \{\text{check nodes connected to variable node } c_n\},$
- $\mathcal{M}(n) \setminus m =$
 $\{\text{check nodes connected to variable node } c_n \text{ excluding check node } f_m\},$
- $M_v(\sim n) = \{\text{messages from all variable nodes except node } c_n\},$
- $M_c(\sim m) = \{\text{messages from all check nodes except node } f_m\},$
- $P_n^b = Pr(c_n = b|y_n),$
- $l_{mn}(b) = Pr(\text{check equation } f_m \text{ is satisfied} | c_n = b, M_v(\sim n)),$
- $z_{nm}(b) = Pr(c_n = b|y_n, M_c(\sim m)), \text{ where } b \in \{0, 1\}.$

Note that the messages $z_{nm}(b)$, while interpreted as probabilities here, are random variables (rv's) as they are functions of y_n 's and other messages which are themselves rv's. Similarly, the messages $l_{mn}(b)$ are rv's.

Consider now the form of $z_{nm}(b)$ which, given our new notation and the independence assumption, we may express as

$$\begin{aligned}
 z_{nm}(b) &= Pr(c_n = b|y_n, M_c(\sim m)) \\
 &= Pr(c_n = b|y_n) Pr(M_c(\sim m) | c_n = b) / Pr(M_c(\sim m)) \\
 &= K_{nm} P_n^b \prod_{m' \in \mathcal{M}(n) \setminus m} l_{m'n}(b), \tag{3.1}
 \end{aligned}$$

3.1 Belief Propagation Algorithm

36

where we used Bayes' rule to obtain the second line and the independence assumption to obtain the third line. The constants K_{nm} are chosen to ensure that $z_{nm}(0) + z_{nm}(1) = 1$.

To develop an expression for the $l_{mn}(b)$, we need the following theorem.

Theorem 3.1. *Consider a sequence of M independent binary digits a_i for which $p_i = \Pr(a_i = 1)$. Then, the probability that $\{a_i | 1 \leq i \leq M\}$ contains an even number of 1's is*

$$\frac{1}{2} + \frac{1}{2} \prod_{i=1}^M (1 - 2p_i). \quad (3.2)$$

In view of this theorem, we have

$$l_{mn}(0) = \frac{1}{2} + \frac{1}{2} \prod_{n' \in \mathcal{N}(m) \setminus n} (1 - 2z_{n'm}(1)) \quad (3.3)$$

since, when $c_n = 0$, the bits $\{c_{n'} | n' \in \mathcal{N}(m) \setminus n\}$ must contain an even number of 1's in order for the check equation f_m to be satisfied. Clearly,

$$l_{mn}(1) = 1 - l_{mn}(0). \quad (3.4)$$

The BP algorithm for the computation of the APP's is initialized by setting $z_{nm}(b) = \Pr(c_n = b | y_n)$ for all m, n for which $h_{mn} = 1$ (i.e., $z_{nm}(b) = P_n^b$). Here, y_n represents channel symbol that was actually received (i.e., it is not a variable here).

Summary of the Probability Domain BP Algorithm

1. For $n = 0, 1, \dots, N - 1$, set $P_n^b = \Pr(c_n = b | y_n)$, where y_n is the n th received channel symbol. Then, set $z_{nm}(b) = P_n^b$ for all m, n for which $h_{mn} = 1$.
2. Update $\{l_{mn}(b)\}$ using (3.3) and (3.4).
3. Update $\{z_{nm}(b)\}$ using (3.1). Solve for the constants K_{nm} .

3.1 Belief Propagation Algorithm

37

4. For $n = 0, 1, \dots, N - 1$, compute

$$z_n(b) = K_n P_n^b \prod_{m \in \mathcal{M}(n)} l_{mn}(b), \quad (3.5)$$

where the constants K_n are chosen to ensure that $z_n(0) + z_n(1) = 1$.

5. For $n = 0, 1, \dots, N - 1$, set

$$\hat{c}_n = \begin{cases} 1 & \text{if } z_n(1) > z_n(0) \\ 0 & \text{else} \end{cases}.$$

If $\hat{\mathbf{c}}\mathbf{H}^t = \mathbf{0}$ or the number of iterations equals the maximum limit, stop; else, go to Step 2.

3.1.2 Log Domain BP Algorithm

The probability domain BP algorithm suffers from the implementation of multiplications (additions are less costly to implement). Besides, many multiplications of probabilities are involved which could become numerically unstable. Thus, a log domain version of the BP algorithm is to be preferred. To do so, we first define the following LLR's:

$$\begin{aligned} F_n &= \log \left(\frac{Pr(c_n = 0|y_n)}{Pr(c_n = 1|y_n)} \right) \\ L_{mn} &= \log \left(\frac{l_{mn}(0)}{l_{mn}(1)} \right) \\ Z_{nm} &= \log \left(\frac{z_{nm}(0)}{z_{nm}(1)} \right) \\ Z_n &= \log \left(\frac{z_n(0)}{z_n(1)} \right) \end{aligned}$$

Next, we obtain the steps of the log domain BP algorithm based on the steps of its probability domain version.

For the initialization step, we should first determine the channel model. We consider the transmission over the AWGN and the uncorrelated Rayleigh fading

3.1 Belief Propagation Algorithm

38

channels with BPSK signaling. Let $\mathbf{c} = (c_0, c_1, \dots, c_{N-1})$ be a codeword, and $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$ be the transmitted signal with the standard mapping $x_n = 1 - 2c_n$ for $0 \leq n \leq N-1$. Then, the received sequence $\mathbf{y} = (y_0, y_1, \dots, y_{N-1})$ for the AWGN channel follows $y_n = x_n + \nu_n$, and for the case of the uncorrelated Rayleigh fading channel, it follows $y_n = a_n x_n + \nu_n$. In these channel models, ν_n 's are independent and identically distributed (i.i.d.) zero-mean Gaussian random variables with variance $N_0/2$, and a_n 's are i.i.d. Rayleigh fading factors with $E[a_n^2] = 1$ and density function $f(a_n) = 2a_n \exp(-a_n^2)$.

For the AWGN channel, we have

$$p(y_n | c_n = b) = \frac{1}{\sqrt{\pi N_0}} \exp \left(-\frac{(y_n - (1 - 2b))^2}{N_0} \right), \quad (3.6)$$

which results in

$$\begin{aligned} Pr(c_n = b | y_n) &= \frac{Pr(c_n = b)p(y_n | c_n = b)}{p(y_n)} \\ &= \frac{Pr(c_n = b)p(y_n | c_n = b)}{Pr(c_n = 0)p(y_n | c_n = 0) + Pr(c_n = 1)p(y_n | c_n = 1)} \\ &= \frac{1}{1 + \exp \left(-\frac{4(1-2b)y_n}{N_0} \right)}, \end{aligned} \quad (3.7)$$

where we assumed equiprobable transmission (i.e., $Pr(c_n = 0) = Pr(c_n = 1)$) in the third line.

Next, F_n is found to be

$$F_n = \log \left(\frac{Pr(c_n = 0 | y_n)}{Pr(c_n = 1 | y_n)} \right) = \frac{4y_n}{N_0}. \quad (3.8)$$

For the case of the uncorrelated Rayleigh fading channel, when we have *ideal channel state information* (CSI), the initial message can be found in a similar manner to the AWGN channel, and is given by

$$F_n = \frac{4y_n a_n}{N_0}, \quad (3.9)$$

3.1 Belief Propagation Algorithm

39

and when no CSI is available, F_n is approximated as

$$F_n \approx \frac{4y_n}{N_0} E[a_n], \quad (3.10)$$

where $E[a_n] = 0.8862$ [16].

Finally, set $Z_{nm} = F_n$ for all m, n for which $h_{mn} = 1$.

For Step 1, we first replace $l_{mn}(0)$ with $1 - l_{mn}(1)$ in (3.3), and rearrange it to obtain

$$1 - 2l_{mn}(1) = \prod_{n' \in \mathcal{N}(m) \setminus n} (1 - 2z_{n'm}(1)). \quad (3.11)$$

Now, using the fact that $\tanh\left[\frac{1}{2}\log(p_0/p_1)\right] = p_0 - p_1 = 1 - 2p_1$, we may rewrite (3.11) as

$$\tanh\left(\frac{1}{2}L_{mn}\right) = \prod_{n' \in \mathcal{N}(m) \setminus n} \tanh\left(\frac{1}{2}Z_{n'm}\right). \quad (3.12)$$

The problem with these expressions is that we are still left with a product and the complex tanh function. We can remedy this as follows. First, factor Z_{nm} into its sign and magnitude:

$$\begin{aligned} Z_{nm} &= \alpha_{nm}\beta_{nm} \\ \alpha_{nm} &= \text{sign}(Z_{nm}) \\ \beta_{nm} &= |Z_{nm}| \end{aligned}$$

so that (3.12) may be rewritten as

$$\tanh\left(\frac{1}{2}L_{mn}\right) = \prod_{n' \in \mathcal{N}(m) \setminus n} \alpha_{n'm} \cdot \prod_{n' \in \mathcal{N}(m) \setminus n} \tanh\left(\frac{1}{2}\beta_{n'm}\right).$$

Figure 3.1: Plot of the $\phi(x)$ function.

We then have

$$\begin{aligned}
 L_{mn} &= \prod_{n'} \alpha_{n'm} \cdot 2 \tanh^{-1} \left(\prod_{n'} \tanh \left(\frac{1}{2} \beta_{n'm} \right) \right) \\
 &= \prod_{n'} \alpha_{n'm} \cdot 2 \tanh^{-1} \log^{-1} \log \left(\prod_{n'} \tanh \left(\frac{1}{2} \beta_{n'm} \right) \right) \\
 &= \prod_{n'} \alpha_{n'm} \cdot 2 \tanh^{-1} \log^{-1} \sum_{n'} \log \left(\tanh \left(\frac{1}{2} \beta_{n'm} \right) \right) \\
 &= \left(\prod_{n' \in \mathcal{N}(m) \setminus n} \alpha_{n'm} \right) \cdot \phi \left(\sum_{n' \in \mathcal{N}(m) \setminus n} \phi(\beta_{n'm}) \right), \tag{3.13}
 \end{aligned}$$

where we have defined

$$\phi(x) = -\log[\tanh(x/2)] = \log \left(\frac{e^x + 1}{e^x - 1} \right), \tag{3.14}$$

and used the fact that $\phi^{-1}(x) = \phi(x)$ when $x > 0$. The function is fairly well behaved, as shown in Fig. 3.1, and hence may be implemented by a look-up table.

For Step 2, we simply divide $z_{nm}(0)$ by $z_{nm}(1)$ (using (3.1)) and take the logarithm of it to obtain

$$Z_{nm} = F_n + \sum_{m' \in \mathcal{M}(n) \setminus m} L_{m'n}. \tag{3.15}$$

Step 3 is similarly modified, so that

$$Z_n = F_n + \sum_{m \in \mathcal{M}(n)} L_{mn}. \quad (3.16)$$

Summary of the Log Domain BP Algorithm

1. For $n = 0, 1, \dots, N - 1$, initialize $Z_{nm} = F_n$ according to (3.8), (3.9), and (3.10) for all m, n for which $h_{mn} = 1$.
2. Update $\{L_{mn}\}$ using (3.13).
3. Update $\{Z_{nm}\}$ using (3.15).
4. Update $\{Z_n\}$ using (3.16).
5. For $n = 0, 1, \dots, N - 1$, set

$$\hat{c}_n = \begin{cases} 1 & \text{if } Z_n < 0 \\ 0 & \text{else} \end{cases}.$$

If $\hat{\mathbf{c}}\mathbf{H}^t = \mathbf{0}$ or the number of iterations equals the maximum limit, stop; else, go to Step 2.

3.2 Min-Sum Algorithm

It should be clear from the previous section that the log domain BP algorithm has lower complexity and is more numerically stable than the probability domain BP algorithm. We now present the min-sum (MS) algorithm which offers a much lower complexity compared with the BP algorithm with the expense of some performance degradation [17].

Consider the update (3.13) for L_{mn} in the log domain decoder. Note from the shape of $\phi(x)$ that the term corresponding to the smallest β_{nm} in the summation

3.2 Min-Sum Algorithm

42

dominates, so that

$$\begin{aligned}\phi\left(\sum_{n'}\phi(\beta_{n'm})\right) &\simeq \phi\left(\phi\left(\min_{n'}\beta_{n'm}\right)\right) \\ &= \min_{n' \in \mathcal{N}(m) \setminus n} \beta_{n'm}.\end{aligned}$$

Thus, the MS algorithm is simply the log domain BP algorithm with Step 1 replaced by

$$L_{mn} = \left(\prod_{n' \in \mathcal{N}(m) \setminus n} \alpha_{n'm} \right) \cdot \min_{n' \in \mathcal{N}(m) \setminus n} \beta_{n'm}. \quad (3.17)$$

It can be shown that for the MS algorithm, if we multiply all the initial messages by a constant number, the hard decisions made at the end of any given iteration will not change. Therefore, it is possible to simplify the initialization step in a way that the knowledge of the Gaussian noise power $N0/2$ becomes unnecessary. Consequently, for the MS algorithm we replace (3.8), (3.9), and (3.10) by

$$F_n = y_n, \quad (3.18)$$

for the case of the AWGN channel and the uncorrelated Rayleigh fading channel without CSI, and

$$F_n = y_n a_n, \quad (3.19)$$

when CSI is available.

Example 3.1. Consider a type-I 2D (4095, 3367) EG-LDPC code. This code is a cyclic code with a rate of 0.82 and a minimum distance of at least 65. The parity-check matrix of this code is a 4095×4095 circulant matrix with row and column weight 64. Fig. 3.2 shows the performance of this code under the BP and the MS algorithms over the AWGN channel. It can be observed that the BER performance of the MS algorithm is more than 1 dB away from that of the BP algorithm. In the next section, we will introduce methods that improve the performance of the MS algorithm. $\square$

Figure 3.2: Bit and block error probabilities of the type-I 2D (4095, 3367) EG-LDPC code based on the BP and the MS algorithms over the AWGN channel.

3.3 Normalized and Offset Min-Sum Algorithm

The computational complexity of the BP algorithm may still be regarded too high for general applications, and there has been an intensive research during the past decade to reduce its complexity while trying to avoid performance degradation. In the previous section, the MS algorithm was introduced as an approximation of the BP algorithm. This decoder has much less computational complexity, but suffers from performance degradation compared with the BP algorithm. Several methods have been proposed to narrow the gap between the performance of the MS and the BP algorithms. The normalized MS (NMS) algorithm was introduced in [18] which significantly improves the performance of the MS algorithm with imposing a negligible increase to its complexity. In [19], offset MS (OMS) algorithm was presented along with the NMS algorithm, and the *density evolution* (DE) technique [15] was deployed to find the optimum parameters of the decoder. Furthermore, the

NMS algorithm was generalized to irregular LDPC codes in [20].

In this section, we propose the combined Normalized and Offset MS (NOMS) algorithm, and we will analyze its performance over the AWGN and the uncorrelated Rayleigh fading channels. The NOMS algorithm is a generic method that includes the NMS and OMS algorithms as special cases. It will be shown that the NOMS algorithm has a better performance than the NMS and OMS algorithms, and its performance approaches that of the BP algorithm.

As previously mentioned, the MS algorithm considerably reduces the computational complexity of the BP algorithm by replacing the tanh function with a simple minimization operation in the step of the check node update. It is proved in [18] that this approximation makes the magnitude of the outgoing LLR's of the check nodes to become larger while retaining their sign unchanged. Thus, they suggest recovering the performance of the MS algorithm by reducing the absolute values of the output LLR's of the check nodes. They introduce the NMS algorithm in which the output LLR's of the check nodes are divided by a normalization factor greater than one. This factor is determined as the ratio of the expected value of the magnitude of LLR's for the MS algorithm over that of the BP algorithm, so that the mean value of the LLR's in the MS algorithm is forced to the average value of the LLR's in the BP algorithm. In [19], the magnitude reduction is achieved by either a normalization factor or an offset term, and the optimum values for these parameters are obtained by the DE technique. The performance of the MS algorithm would be the same as the BP algorithm if the probability density functions (pdf's) of the LLR's were the same for both algorithms. The normalization of the LLR's can be viewed as a scaling of their pdf's whereas reducing the magnitude of the LLR's by an offset term can be viewed as a shift of the pdf's. The pdf's of the LLR's in the NMS and OMS algorithms approach their corresponding pdf's in the BP algorithm

by the scaling or shifting operations. This makes an intuition that the pdf's in the MS algorithm may better match those in the BP algorithm if both scaling and shifting operations are deployed. Thus, we propose to combine the NMS and OMS algorithms to take advantage of both scaling and shifting simultaneously.

The NOMS algorithm has the same steps as the MS algorithm with an extra operation after the check node update:

$$L_{mn} \leftarrow \alpha \cdot \text{sgn}(L_{mn}) \cdot \max(|L_{mn}| - \beta, 0), \quad (3.20)$$

where $0 < \alpha \leq 1$ and $\beta \geq 0$ are the normalization factor and offset term, respectively. For the sake of simplicity, we assume the (α, β) pair to be constant for all iterations and different SNR values. The (α, β) pair can be optimized for each node degree of the Tanner graph in the case of an irregular LDPC code [20].

The NOMS algorithm is a generic method that includes the NMS and OMS algorithms as special cases. For the $(\alpha, 0)$ pair, the NOMS algorithm degenerates to the NMS algorithm, and for the $(1, \beta)$ pair, the NOMS algorithm degenerates to the OMS algorithm.

The NMS, OMS, and NOMS algorithms have almost the same order of complexity. Take note that α and β are determined offline just for one time, and they do not need to be updated while the decoding algorithm is running. Therefore, the *real time* complexity of the algorithm is not affected by these parameters.

3.3.1 Simulation Results

To investigate the performance of our proposed NOMS decoder, we used a type-I 2-D (1023, 781) EG-LDPC code for our simulations. The parity-check matrix of this code is circulant, and therefore, this code is cyclic. It is a (32, 32)-regular LDPC code, and the length of the shortest cycle in its Tanner graph is 6. This code has a minimum distance of at least 33 [6].

3.3 Normalized and Offset Min-Sum Algorithm

46

Figure 3.3: Bit error probabilities of the (1023, 781) EG-LDPC code over the AWGN channel.

Fig. 3.3 shows the bit error probabilities of the (1023, 781) EG-LDPC code with the BP, NMS, OMS, and NOMS algorithms over the AWGN channel. The maximum number of iterations is set to 50 for all the simulations. The optimum value of the normalization factor for the NMS algorithm is 0.3, and the optimum value of the offset term for the OMS algorithm is 0.2. These values have been found by simulation. The NMS and OMS algorithms have a BER performance close to each other, and it is observed that there is a gap of about 0.06 dB between the BER performance of the NMS algorithm and that of the BP algorithm at the bit error probability of 10^{-5} . The optimum value of the (α, β) pair for the NOMS algorithm is found to be (0.45, 0.08) by simulation. The proposed algorithm achieves a coding gain of 0.04 dB over the NMS algorithm, and is 0.02 dB away from the BER performance of the BP algorithm at the bit error probability of 10^{-5} .

Fig. 3.4 and Fig. 3.5 illustrate the bit error probabilities of the LDPC code under study over the uncorrelated Rayleigh fading channel with ideal CSI and without CSI,

3.3 Normalized and Offset Min-Sum Algorithm

47

Figure 3.4: Bit error probabilities of the (1023, 781) EG-LDPC code over the uncorrelated Rayleigh fading channel with ideal CSI.

Figure 3.5: Bit error probabilities of the (1023, 781) EG-LDPC code over the uncorrelated Rayleigh fading channel without CSI.

respectively. The same observations can be made for the BER performance of the code over the Rayleigh fading channel as it was made for the AWGN channel. Note that the optimum values of the parameters are different from those for the AWGN channel. The optimum values of the (α, β) pair for the uncorrelated Rayleigh fading channel with ideal CSI and without CSI are found to be $(0.6, 0.03)$ and $(0.55, 0.08)$ respectively. The proposed algorithm exhibits a better improvement in the BER performances for the Rayleigh fading channel compared with the AWGN channel. As depicted in Fig. 3.5, the NOMS algorithm achieves 0.15 dB coding gain over the NMS algorithm and is only 0.04 dB away from the BER performance of the BP algorithm at the bit error probability of 10^{-5} . This result suggests that for the Rayleigh fading channel, a better matching of the pdf of the LLR's is obtained by both scaling and shifting operations.

Chapter 4

Detection and Decoding Over Partial Response Channels

In this chapter, we investigate the performance of the NOMS algorithm over partial response (PR) channels, and we show that it outperforms the NMS and OMS algorithms. First, we introduce the channel model, and next we describe the turbo decoding algorithm which is an iterative method used for the purpose of detection and decoding.

A coded PR channel can be represented as a serial concatenated system, where an LDPC encoder precedes a finite impulse response (FIR) filter representing the inter-symbol interference (ISI) caused by the channel memory. First, K user data bits $\mathbf{b} = (b_0, b_1, \dots, b_{K-1})$ are encoded by an LDPC encoder into a codeword $\mathbf{c} = (c_0, c_1, \dots, c_{N-1})$ of length N . Next, the transmitted sequence $\mathbf{x} = (x_0, x_1, \dots, x_{N-1})$ follows the bipolar signalling, i.e. $x_n = 2c_n - 1$ ($x_n \in \{-1, +1\}$), and passes over an ideal channel whose impulse response is denoted as $\{h_i\}, i = 0, 1, \dots, L$, where L is the length of the channel memory. The received sequence $\mathbf{y} = (y_0, y_1, \dots, y_{N-1})$ is

Figure 4.1: Turbo decoding scheme for coded PR channels.

given by

$$y_n = \sum_{i=0}^L h_i x_{n-i} + \nu_n, \quad (4.1)$$

where ν_n 's are independent and identically distributed (i.i.d.) zero-mean Gaussian random variables with variance $N_0/2$.

Detection and decoding is performed in an iterative manner as illustrated in Fig. 4.1. This scheme is known as turbo decoding, and comprises a soft-in soft-out (SISO) detector and a SISO decoder. The detector and decoder exchange *extrinsic* soft information as shown in Fig. 4.1. In this figure, L_n^{det} and L_n^{dec} represent the *extrinsic* information produced by the detector and the decoder respectively. Take note that our bipolar signalling used for the detector is different from the signalling used for the decoder in the previous chapter. This type of mapping makes our formulation simpler. We can use different signalling for the detector and the decoder simultaneously. The only thing we need to do is to multiply the output messages of the detector (or decoder) by “ -1 ” before feeding the decoder (or detector).

In the next section, we describe the details of the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm which is the optimal method for channel detection [21].

4.1 BCJR Channel Detector

In this section, we elaborate the derivation of the BCJR algorithm for PR channels both in probability and log domains [21]. The goal is to compute the LLR

$$L(c_n|\mathbf{y}) = \log \left(\frac{Pr(c_n = 0|\mathbf{y})}{Pr(c_n = 1|\mathbf{y})} \right), \quad (4.2)$$

given the received sequence $\mathbf{y}$.

4.1.1 Probability Domain BCJR Algorithm

In order to incorporate the channel trellis into the computation of $L(c_n|\mathbf{y})$, we rewrite (4.2) as

$$L(c_n|\mathbf{y}) = \log \frac{\sum_{C^0} p(s_{n-1} = s', s_n = s, \mathbf{y})}{\sum_{C^1} p(s_{n-1} = s', s_n = s, \mathbf{y})} \quad (4.3)$$

where $s_n = (c_n, c_{n-1}, \dots, c_{n-L+1})$ is the channel state at time instant n , C^0 is set of pairs (s', s) for the state transitions $(s_{n-1} = s') \rightarrow (s_n = s)$ which correspond to the event $c_n = 0$, and C^1 is similarly defined. To write (4.3) we used Bayes' rule, total probability, and then canceled $1/p(\mathbf{y})$ in the numerator and denominator. We see from (4.3) that we only need to compute $p(s', s, \mathbf{y}) = p(s_{n-1} = s', s_n = s, \mathbf{y})$ for all state transitions and then sum over the appropriate transitions in the numerator and denominator. We now provide the facts which facilitate the computation of $p(s', s, \mathbf{y})$.

Fact 4.1. The pdf $p(s', s, \mathbf{y})$ may be factored as

$$p(s', s, \mathbf{y}) = \alpha_{n-1}(s') \cdot \gamma_n(s', s) \cdot \beta_n(s), \quad (4.4)$$

where

$$\begin{aligned} \alpha_n(s) &\triangleq p(s_n = s, \mathbf{y}_0^n) \\ \gamma_n(s', s) &\triangleq p(s_n = s, y_n | s_{n-1} = s') \\ \beta_n(s) &\triangleq p(\mathbf{y}_{n+1}^{N-1} | s_n = s), \end{aligned}$$

4.1 BCJR Channel Detector

52

and where $\mathbf{y}_a^b = (y_a, y_{a+1}, \dots, y_b)$.

Proof. By several applications of Bayes' rule, we have

$$\begin{aligned}
 p(s', s, \mathbf{y}) &= p(s', s, \mathbf{y}_0^{n-1}, y_n, \mathbf{y}_{n+1}^{N-1}) \\
 &= p(\mathbf{y}_{n+1}^{N-1} | s', s, \mathbf{y}_0^{n-1}, y_n) p(s', s, \mathbf{y}_0^{n-1}, y_n) \\
 &= p(\mathbf{y}_{n+1}^{N-1} | s', s, \mathbf{y}_0^{n-1}, y_n) p(s, y_n | s', \mathbf{y}_0^{n-1}) p(s', \mathbf{y}_0^{n-1}) \\
 &= p(\mathbf{y}_{n+1}^{N-1} | s) p(s, y_n | s') p(s', \mathbf{y}_0^{n-1}) \\
 &= \beta_n(s) \cdot \gamma_n(s', s) \cdot \alpha_{n-1}(s'),
 \end{aligned}$$

where the fourth line follows from the third because the variables omitted on the fourth line are conditionally independent. $\square$

Fact 4.2. The probability $\alpha_n(s)$ may be computed in a “forward recursion” via

$$\alpha_n(s) = \sum_{s'} \gamma_n(s', s) \alpha_{n-1}(s'), \quad (4.5)$$

where the sum is over all possible channel states.

Proof. By several applications of Bayes' rule and the theorem on total probability, we have

$$\begin{aligned}
 \alpha_n(s) &= p(s, \mathbf{y}_0^n) \\
 &= \sum_{s'} p(s', s, \mathbf{y}_0^n) \\
 &= \sum_{s'} p(s, y_n | s', \mathbf{y}_0^{n-1}) p(s', \mathbf{y}_0^{n-1}) \\
 &= \sum_{s'} p(s, y_n | s') p(s', \mathbf{y}_0^{n-1}) \\
 &= \sum_{s'} \gamma_n(s', s) \alpha_{n-1}(s'),
 \end{aligned}$$

where the fourth line follows from the third due to conditional independence of $\mathbf{y}_0^{n-1}$. $\square$

Fact 4.3. The probability $\beta_n(s)$ may be computed in a “backward recursion” via

$$\beta_{n-1}(s') = \sum_s \beta_n(s) \gamma_n(s', s). \quad (4.6)$$

Proof. By applying the Bayes’ rule and the theorem on total probability, we have

$$\begin{aligned} \beta_{n-1}(s') &= p(\mathbf{y}_n^{N-1} | s') \\ &= \sum_s p(\mathbf{y}_n^{N-1}, s | s') \\ &= \sum_s p(\mathbf{y}_{n+1}^{N-1} | s', s, y_n) p(s, y_n | s') \\ &= \sum_s p(\mathbf{y}_{n+1}^{N-1} | s) p(s, y_n | s') \\ &= \sum_s \beta_n(s) \gamma_n(s', s), \end{aligned}$$

where conditional independence led to the omission of variables on the forth line. $\square$

The recursion for the $\{\alpha_n(s)\}$ is initialized according to

$$\alpha_{-1}(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases},$$

which makes the reasonable assumption that the channel state is initialized to zero.

The recursion for the $\{\beta_n(s)\}$ is initialized according to

$$\beta_{N-1}(s) = \begin{cases} 1, & s = \mathbf{0} \\ 0, & s \neq \mathbf{0} \end{cases},$$

which assumes that “ L termination zero bits” have been included at the end of the codeword $\mathbf{c}$, so that the channel is again in the zero state at time $N - 1$.

All that remains at this point is the computation of $\gamma_n(s', s) = p(s, y_n | s')$. Observe that $\gamma_n(s', s)$ may be written as

$$\begin{aligned} \gamma_n(s', s) &= \frac{Pr(s', s)}{Pr(s')} \cdot \frac{p(s', s, y_n)}{Pr(s', s)} \\ &= Pr(s | s') p(y_n | s', s). \end{aligned} \quad (4.7)$$

4.1 BCJR Channel Detector

54

Note that $Pr(s|s')$ and subsequently $\gamma_n(s', s)$ is equal to zero in case that there is no possible transition from state s' to state s , and $Pr(s|s') = Pr(c_n)$ for a valid transition. Hence $\gamma_n(s', s) = 0$ if $s' \rightarrow s$ is not valid, and otherwise

$$\gamma_n(s', s) = \frac{Pr(c_n)}{\sqrt{2\pi\sigma^2}} \exp \left[-\frac{(y_n - z_n)^2}{2\sigma^2} \right], \quad (4.8)$$

where $\sigma^2 = N_0/2$, and z_n is the noiseless output of the channel obtained by

$$z_n = \sum_{i=0}^L h_i x_{n-i}. \quad (4.9)$$

Note that prior to the execution of the algorithm, z_n can be computed for all possible sequences of length $L + 1$, and these values can be stored in a look-up table. In (4.8), $Pr(c_n)$ represents the *prior* information about c_n , and it is fed to the detector by the decoder. For the first turbo iteration, where there is no *prior* information available for the detector, a uniform distribution for $Pr(c_n)$ is deployed.

In (4.3), the computation of the numerator or denominator can be simplified as

$$\begin{aligned} \sum_{C^b} p(s', s, \mathbf{y}) &= \sum_{(s', s)|c_n=b} \beta_n(s) \gamma_n(s', s) \alpha_{n-1}(s') \\ &= \sum_{s|c_n=b} \left(\beta_n(s) \sum_{s'} \gamma_n(s', s) \alpha_{n-1}(s') \right) \\ &= \sum_{s|c_n=b} \beta_n(s) \alpha_n(s), \end{aligned} \quad (4.10)$$

where $b \in \{0, 1\}$, and $s|c_n = b$ represents the states for which c_n equals b .

In practice, the BCJR algorithm is implemented in log domain to achieve more numerical stability and less complexity.

4.1.2 Log Domain BCJR Algorithm

In the log domain BCJR algorithm, $\alpha_n(s)$ is replaced by the *forward metric*

$$\begin{aligned}\tilde{\alpha}_n(s) &\triangleq \log(\alpha_n(s)) \\ &= \log\left(\sum_{s'} \alpha_{n-1}(s') \gamma_n(s', s)\right) \\ &= \log\left(\sum_{s'} \exp(\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s))\right),\end{aligned}\quad (4.11)$$

where the *branch metric* $\tilde{\gamma}_n(s', s)$ is given by

$$\begin{aligned}\tilde{\gamma}_n(s', s) &\triangleq \log(\gamma_n(s', s)) \\ &= \log(Pr(c_n)) - \log\left(\sqrt{2\pi\sigma^2}\right) - \frac{(y_n - z_n)^2}{2\sigma^2}.\end{aligned}\quad (4.12)$$

Let L_n^{dec} be the *prior* information of the n th symbol defined by

$$L_n^{dec} \triangleq \log\left(\frac{Pr(c_n = 0)}{Pr(c_n = 1)}\right), \quad (4.13)$$

which is fed back from the decoder into the BCJR detector. Then, (4.12) can be rewritten as

$$\begin{aligned}\tilde{\gamma}_n(s', s) &= -c_n L_n^{dec} + L_n^{dec} - \log(1 + \exp(L_n^{dec})) - \log\left(\sqrt{2\pi\sigma^2}\right) - \frac{(y_n - z_n)^2}{2\sigma^2} \\ &= -c_n L_n^{dec} - \frac{(y_n - z_n)^2}{2\sigma^2} + d_n,\end{aligned}\quad (4.14)$$

where $c_n \in \{0, 1\}$, and d_n is a function of n , and its value does not depend on the state transition $s' \rightarrow s$. Later, we show that d_n can be omitted from the computation of $\tilde{\gamma}_n(s', s)$. For the first turbo iteration, the *prior* information L_n^{dec} is set to zero.

Note that (4.11) not only defines $\tilde{\alpha}_n(s)$, but it gives its recursion. These log domain forward metrics are initialized as

$$\tilde{\alpha}_{-1}(s) = \begin{cases} 0, & s = \mathbf{0} \\ -\infty, & s \neq \mathbf{0} \end{cases}.$$

The probability $\beta_{n-1}(s')$ is replaced by the *backward metric*

$$\begin{aligned}\tilde{\beta}_{n-1}(s') &\triangleq \log(\beta_{n-1}(s')) \\ &= \log \left(\sum_s \exp(\tilde{\beta}_n(s) + \tilde{\gamma}_n(s', s)) \right)\end{aligned}\quad (4.15)$$

with initial conditions

$$\tilde{\beta}_{N-1}(s) = \begin{cases} 0, & s = \mathbf{0} \\ -\infty, & s \neq \mathbf{0} \end{cases}$$

under the assumption that the channel has been terminated to the zero state.

With respect to (4.10), (4.3) can be rewritten as

$$\begin{aligned}L(c_n|\mathbf{y}) &= \log \left(\sum_{s|c_n=0} \alpha_n(s) \beta_n(s) \right) - \log \left(\sum_{s|c_n=1} \alpha_n(s) \beta_n(s) \right) \\ &= \log \left(\sum_{s|c_n=0} \exp(\tilde{\alpha}_n(s) + \tilde{\beta}_n(s)) \right) \\ &\quad - \log \left(\sum_{s|c_n=1} \exp(\tilde{\alpha}_n(s) + \tilde{\beta}_n(s)) \right)\end{aligned}\quad (4.16)$$

As mentioned before, the term d_n can be omitted from (4.14). From (4.11) and (4.15), it is straightforward to show that the omission of d_n from $\tilde{\gamma}_n(s', s)$ results in an additive term in the computation of $\tilde{\alpha}_n(s)$ and $\tilde{\beta}_n(s)$. In other words, if we replace $\tilde{\gamma}_n(s', s)$ by $\tilde{\gamma}_n(s', s) - d_n$, we obtain $\tilde{\alpha}_n(s) - d'_n$ and $\tilde{\beta}_n(s) - d''_n$ instead of getting $\tilde{\alpha}_n(s)$ and $\tilde{\beta}_n(s)$ respectively. Noting that the terms d'_n and d''_n are independent of the channel state s , it can be easily shown that replacing $\tilde{\alpha}_n(s)$ and $\tilde{\beta}_n(s)$ by $\tilde{\alpha}_n(s) - d'_n$ and $\tilde{\beta}_n(s) - d''_n$ does not effect the computation of $L(c_n|\mathbf{y})$ in (4.16). Therefore, there is no need for the computation of the term d_n in (4.14).

At first glance, log domain equations do not look any simpler than the probability domain algorithm, but we use the following facts to attain the simplification.

Fact 4.4.

$$\max(u, v) = \log \left(\frac{e^u + e^v}{1 + e^{-|u-v|}} \right).$$

4.1 BCJR Channel Detector

57

Proof. Without loss of generality, when $u > v$, the right-hand side equals u . $\square$

Now define

$$\max^*(u, v) \triangleq \log(e^u + e^v), \quad (4.17)$$

so that from Fact 4.4,

$$\max^*(u, v) = \max(u, v) + \log(1 + e^{-|u-v|}) \quad (4.18)$$

This may be extended to more than two variables. For example,

$$\max^*(u, v, w) \triangleq \log(e^u + e^v + e^w),$$

which may be computed pairwise according to the following fact.

Fact 4.5.

$$\max^*(u, v, w) = \max^*(\max^*(u, v), w)$$

Proof.

$$\begin{aligned} \max^*(\max^*(u, v), w) &= \log(e^{\max^*(u, v)} + e^w) \\ &= \log(e^{\log(e^u + e^v)} + e^w) \\ &= \log(e^u + e^v + e^w) \\ &= \max^*(u, v, w) \end{aligned}$$

$\square$

Given the function $\max^*(\cdot)$, we may now rewrite (4.11), (4.15), and (4.16) as

$$\tilde{\alpha}_n(s) = \max_{s'}^* (\tilde{\alpha}_{n-1}(s') + \tilde{\gamma}_n(s', s)), \quad (4.19)$$

$$\tilde{\beta}_{n-1}(s') = \max_s^* (\tilde{\beta}_n(s) + \tilde{\gamma}_n(s', s)), \quad (4.20)$$

and

$$\begin{aligned} L(c_n | \mathbf{y}) &= \max_{s|c_n=0}^* (\tilde{\alpha}_n(s) + \tilde{\beta}_n(s)) \\ &\quad - \max_{s|c_n=1}^* (\tilde{\alpha}_n(s) + \tilde{\beta}_n(s)). \end{aligned} \quad (4.21)$$

We see from (4.19), (4.20), and (4.21) how the log domain computation of $L(c_n|\mathbf{y})$ is vastly simplified relative to the probability domain computation. From (4.18), implementation of the $\max^*(\cdot)$ function involves only a two-input $\max(\cdot)$ function plus a look-up table for the “correction term” $\log(1 + e^{-|u-v|})$. Note that the correction term is bounded as

$$0 < \log(1 + e^{-|u-v|}) \leq \log(2) \simeq 0.693,$$

so that $\max^*(u, v) \simeq \max(u, v)$ when $|\max(u, v)| \geq 7$.

Finally, the output *extrinsic* information of the BCJR detector is obtained by

$$L_n^{det} = L(c_n|\mathbf{y}) - L_n^{dec}, \quad (4.22)$$

and is passed to the LDPC decoder.

4.2 NOMS Algorithm Over PR Channels

In Chapter 3, we proposed the NOMS algorithm for decoding LDPC codes, and we analyzed its performance over the AWGN and the uncorrelated Rayleigh fading channels. In this section, we investigate the performance of the NOMS algorithm over PR channels. For PR channels, the NOMS algorithm is initialized by the *extrinsic* information obtained from the detector, i.e.,

$$F_n = L_n^{det}. \quad (4.23)$$

The other steps of the NOMS algorithm are the same as described in Chapter 3.

After the last iteration of the decoder, the *extrinsic* information is obtained by

$$L_n^{dec} = Z_n - L_n^{det}, \quad (4.24)$$

where Z_n was defined in (3.16), and finally, L_n^{dec} is fed back to the detector.

4.3 Simulation Results

In this section, we investigate the decoding performance of the (1023, 781) EG-LDPC code with the BP, NMS, OMS, and NOMS algorithms over the MEEPR4 (modified extended-extended partial response class 4) channel whose coefficients are given by $\mathbf{h} = \{5, 4, -3, -4, -2\}$ [22, 23]. Channel detection is performed by the BCJR algorithm. We use 3 turbo iterations between the BCJR detector and the LDPC decoder, and the maximum number of iterations for the LDPC decoder is set to 30 in our simulations.

Fig. 4.2 shows the simulation results. The optimum value of the normalization factor for the NMS algorithm is 0.2, and the optimum value of the offset term for the OMS algorithm is 1.4. These parameters have been determined using simulation. The NMS and OMS algorithms have a BER performance very close to each other up to the bit error probability of 10^{-4} , and then the OMS algorithm starts to approach an error floor for higher SNR values. It is observed that there is a gap of about 0.15 dB between the BER performance of the NMS algorithm and that of the BP algorithm at the bit error probability of 10^{-4} . The optimum value of the (α, β) pair for the NOMS algorithm is found to be (0.5, 0.9). These parameters have been obtained by simulation. The NOMS algorithm achieves a coding gain of 0.07 dB over the NMS algorithm, and 0.12 dB over the OMS algorithm at the bit error probability of 10^{-4} .

In Fig. 4.2, similar observations can be made for the block error probabilities as it was made for the bit error probabilities. It is shown that the NOMS algorithm outperforms the NMS and OMS algorithms in terms of the block error probability as well.

4.3 Simulation Results

60

Figure 4.2: Bit and block error probabilities of the (1023, 781) EG-LDPC code over the MEEPR4 channel.

Chapter 5

Markov Chain Monte Carlo Based Detection Over 2-D Magnetic Recording Channels

We start this chapter by introducing two-dimensional (2-D) channels, and we focus our attention on the problem of channel detection over these type of systems. A popular and important instance of this class of channels is the 2-D inter-symbol interference (ISI) channel, which appears, e.g., in magnetic and optical recording devices [24]. In these recording systems, there have been proposals to increase the capacity by reducing inter-track spacing. However, such dense spacing of tracks leads to not just the conventional down-track ISI, but to cross-track ISI too, which together forms 2-D ISI. For example, in 2-D optical recording and bit-patterned magnetic recording (BPMR) systems, data is stored in a 2-D array, and naturally suffers from 2-D ISI.

In contrast to 1-D channels, where optimal detection is applicable using the BCJR and Viterbi algorithms, there does not exist a practical implementation of an optimal detection method for 2-D channels because of the prohibitive compu-

tational complexity. It has been proved in [25] that maximum-likelihood sequence detection for 2-D channels is NP-complete. Besides, the BCJR algorithm for 2-D channels has a computational complexity that grows exponentially not only with the size of the channel coefficients but also with the width of the transmitted data blocks. Consequently, the implementation of optimal algorithms for 2-D channels is impractical, and sub-optimal solutions need to be considered.

In the following sections, we will briefly review the existing methods. Although sub-optimal detection algorithms have much lower complexity compared with optimal methods, the computational burden of the detection step is still the bottleneck of the receiver side. In this chapter, we will propose an algorithm based on Monte Carlo techniques that significantly reduces the computational complexity of the 2-D channel detectors.

5.1 System Model for 2-D ISI Channels

Discrete-input 2-D Gaussian channels with memory represent an important class of systems, which appear extensively in communications and storage. In such systems, a coded information symbol, chosen from a finite alphabet, is impaired by other transmitted symbols and additive white Gaussian noise (AWGN).

The system model for 2-D ISI channels is similar to the model for the coded partial response channels. First, K user data bits $\mathbf{b} = (b_0, b_1, \dots, b_{K-1})$ are encoded by an LDPC encoder into a codeword $\mathbf{c} = (c_0, c_1, \dots, c_{N_c-1})$ of length N_c . Next, the codeword is split into M sequences of length N , where $M \times N$ equals N_c . Now, these M sequences are arranged as the rows of a $M \times N$ matrix. Then, the transmitted matrix $\mathbf{X} = \left\{ x_{m,n} \in \{-1, +1\} \right\}$ is formed by the bipolar signalling as

$$x_{m,n} = 2c_{(n+mN)} - 1 \quad (5.1)$$

5.1 System Model for 2-D ISI Channels

63

for $0 \leq m \leq M - 1$ and $0 \leq n \leq N - 1$. For storage systems, $\mathbf{X}$ is written on the surface of the recording media such that (m, n) pairs represent the coordinates of the symbols. We assume that each block of data is surrounded by “-1” symbols. These zero bits guarantee the independence of the adjacent data blocks, and they are used for the purpose of initialization and termination in channel detection algorithms.

For $0 \leq i \leq L_M$ and $0 \leq j \leq L_N$, let $\{h_{i,j}\}$ denote the coefficients of the discrete channel impulse response, where L_M and L_N represent the channel memory length in the vertical and horizontal directions respectively. Then, the elements of the received matrix $\mathbf{Y} = \{y_{m,n}\}$ for $0 \leq m \leq M + L_M - 1$ and $0 \leq n \leq N + L_N - 1$ are given by the following model:

$$g_{m,n}(\mathbf{X}_{m-L_M:m,n-L_N:n}) = \sum_{i=0}^{L_M} \sum_{j=0}^{L_N} h_{i,j} x_{m-i,n-j}, \quad (5.2)$$

$$y_{m,n} = g_{m,n}(\mathbf{X}_{m-L_M:m,n-L_N:n}) + \nu_{m,n}, \quad (5.3)$$

where $\mathbf{X}_{m-L_M:m,n-L_N:n}$ denotes $\{x_{r,q}\}$ for $m - L_M \leq r \leq m$ and $n - L_N \leq q \leq n$. Further, $g_{m,n}(\cdot)$ represents the noiseless channel output, and $\nu_{m,n}$'s are independent and identically distributed (i.i.d.) zero-mean Gaussian random variables with variance $\sigma^2 = N_0/2$. Note that the symbols that fall outside the block of data in (5.2) are assumed to be “-1”. The number of “-1” symbols required depends on the channel memory length, and there should be at least $(L_M + M + L_M)(L_N + N + L_N) - MN$ number of “-1” symbols around the data block.

Detection and decoding is performed by turbo decoding as it was illustrated in Fig. 4.1. Note that the LDPC decoder is the same as described in previous chapters, and therefore, the output of the detector, which is in matrix format, should be first converted into a vector before being fed to the decoder. This is done by placing the rows of the matrix one after the other such that a sequence is formed. Accordingly, the vector output of the LDPC decoder should be split into a matrix before being fed to the detector.

In the next section, we will briefly review the existing 2-D ISI channel detection methods which are soft-in soft-output, and hence, applicable to the turbo decoding scheme.

5.2 Detection methods over 2-D ISI Channels

In recent years, various sub-optimal detection methods have been proposed for 2-D ISI channels which range from using simple 2-D minimum mean-squared-error (MMSE) equalizers to the use of more sophisticated ideas, like message passing algorithms [26–35].

In [27], MMSE filters are applied to holographic storage. The advantage of linear filtering is its reasonably low complexity. However, these techniques can suffer from noise enhancement, particularly on channels with nulls. In [29], for cases that the matrix of the channel coefficients can be written as the multiplication of a column vector by a row vector (noted as mask separability), an iterative row-column detector is proposed for LDPC coded AWGN channels. In this algorithm, *extrinsic* information is exchanged between a non-binary column SISO detector, a binary row SISO detector, and an LDPC decoder. In [31], soft information is exchanged between BCJR detectors operating on multiple rows and multiple columns of the channel output matrix; this scheme, also known as the iterative multi-strip (IMS) algorithm, handles both separable and non-separable masks.

In [32, 33], the generalized belief propagation (GBP) algorithm [36] is applied for the purpose of detection over 2-D ISI channels. It can be shown that the direct implementation of the belief propagation algorithm results in a graph that contains a large number of cycles of length 4, and therefore, the performance is not expected to be well. To overcome this problem, the generalized belief propagation algorithm

can be used. In this algorithm, nodes of the corresponding factor graph are grouped into clusters such that the obtained regions contain the short cycles. Then, exact inference is performed within each region, and messages are passed between regions instead of the nodes. This algorithm has been shown to have a performance very close to optimal detection [32, 33].

There is another approach called the iterative row-column soft-decision feedback algorithm (IRCSDFA) [34, 35]. The main idea behind the IRCSDFA, IMS, and GBP algorithms is the same, and involves passing messages between locally optimal detectors. These algorithms have similar performance and computational complexity, and therefore, we only consider the IRCSDFA which slightly performs better than the other algorithms [34, 35].

The block diagram of the IRCSDFA is depicted in Fig. 5.1. This algorithm consists of a row-by-row and a column-by-column detector. These detectors exchange *extrinsic* information for a given number of iterations prior to passing the information to the LDPC decoder. At each iteration inside the detector, the soft information is scaled down by the factor w which is normally a function of the iteration number. For the row-by-row detector, first the M -by- N field is split into overlapping horizontal strips as illustrated in Fig. 5.2. The width of each strip is normally greater than the channel memory length in the vertical direction (L_M). Next, the BCJR algorithm is performed for the topmost strip, and then, the obtained soft information (a posteriori probabilities) are passed to the next strip. This procedure goes on to the last strip.

The column-by-column detector is similar to the row-by-row detector with the difference that it employs the BCJR algorithm on vertical strips.

In the next section, we provide the mathematical preliminaries of Monte Carlo methods. Later, we will use these techniques to derive the formulas and steps of our

Figure 5.1: IRCSDFA block diagram.

Figure 5.2: Splitting the M -by- N field into overlapping horizontal strips.

proposed algorithm.

5.3 Monte Carlo Statistical Methods

Monte Carlo techniques provide a set of powerful tools for approximating integrations and summations by replacing an entire distribution by a number of samples drawn from an appropriate distribution [37, 38]. Particularly, these methods can be used to approximate multi-dimensional integrations to avoid a complexity that is exponential in the number of dimensions involved. We are especially interested in this property, as it helps us to derive a solution for approximating *a posteriori* probabilities.

Consider the problem of evaluating the expected value of a function $h(\mathbf{z})$ of the random variable $\mathbf{z}$, given the distribution $f(\mathbf{z})$. In general, $\mathbf{z}$ can be a multi-dimensional random variable. Our goal is to determine

$$E\{h(\mathbf{z})\} = \int_{\mathcal{D}} h(\mathbf{z})f(\mathbf{z})d\mathbf{z}, \quad (5.4)$$

where $\mathcal{D}$ is the domain of $\mathbf{z}$.

5.3.1 Monte Carlo Integration

The classical Monte Carlo integration is a method used to make an estimation of $E\{h(\mathbf{z})\}$. To do so, we first draw N_s samples from $f(\mathbf{z})$, say $\{\mathbf{z}^{(k)}\}_{k=1}^{N_s}$, and then, we compute

$$\bar{h} = \frac{1}{N_s} \sum_{k=1}^{N_s} h(\mathbf{z}^{(k)}). \quad (5.5)$$

We consider $\bar{h}$ as an estimation of $E\{h(\mathbf{z})\}$ since

$$\begin{aligned} E\{\bar{h}\} &= E\left\{\frac{1}{N_s} \sum_{k=1}^{N_s} h(\mathbf{z}^{(k)})\right\} \\ &= E\{h(\mathbf{z})\}. \end{aligned}$$

Furthermore, when the samples are drawn independently, the variance of $\bar{h}$ decreases inversely proportional with N_s , and we have $\text{var}\{\bar{h}\} = \text{var}\{h(\mathbf{z})\}/N_s$. When the samples are correlated, the variance still decreases with N_s , but does so at a slower rate.

Numerical studies show that the dimension of $\mathbf{z}$ and the size of N_s are very weakly related in the sense that even though the dimension of $\mathbf{z}$ may increase, the order of magnitude of N_s remains unchanged. Hence, the exponential growth of the computational complexity that is commonly encountered in performing multiple integrals may be avoided by using the Monte Carlo methods [37, 39].

5.3.2 Markov Chain Monte Carlo (MCMC) Based Sampling

The Monte Carlo methods discussed above require drawing samples from a target distribution $q(\mathbf{z})$. When the dimensionality of $\mathbf{z}$ is high, drawing such samples may still be difficult. In order to alleviate this problem, sampling methods based on Markov chains can be used. Particularly, we will consider the well-known Gibbs sampler which draws samples from a target distribution based on the concept of Markov chains [37, 40].

Before getting into the details of the Gibbs sampler, we will review some concepts of Markov chains and invariant distributions. Then, we show how sampling from a Markov chain is performed, and finally, we will describe the Gibbs sampler [39].

Markov Chains

Consider a finite domain $\mathcal{Z} = \{\mathbf{a}_1, \dots, \mathbf{a}_{N_z}\}$, containing N_z elements. A Markov chain is a sequence of random variables $\mathbf{z}_0, \mathbf{z}_1, \dots$ over $\mathcal{Z}$, satisfying the Markov property

$$p(\mathbf{z}_n | \mathbf{z}_0, \dots, \mathbf{z}_{n-1}) = p(\mathbf{z}_n | \mathbf{z}_{n-1}). \quad (5.6)$$

The set $\mathcal{Z}$ is known as the *state space*, while $\mathbf{z}_n$ is the *state* at time instant $n, n \geq 0$. A Markov chain is homogeneous when the transition probabilities $p(\mathbf{z}_n | \mathbf{z}_{n-1})$ do not depend on the time instant n . In a homogeneous Markov chain, we can define a so-called *transition kernel*

$$Q(\mathbf{z} | \mathbf{z}') = p(\mathbf{z}_n | \mathbf{z}_{n-1}). \quad (5.7)$$

We will write the distribution of the state of the system at time n as $p_n(\cdot)$. Suppose that we are given an initial distribution $p_0(\cdot)$, then the distribution of the state at time $n = 1$ is related to the distribution of the state at time 0 by

$$p_1(\mathbf{z}) = \sum_{\mathbf{z}' \in \mathcal{Z}} Q(\mathbf{z} | \mathbf{z}') p_0(\mathbf{z}'). \quad (5.8)$$

Let us represent the distribution $p_n(\cdot)$ as an $N_z \times 1$ column-vector $\mathbf{p}_n = \{p_n(\mathbf{a}_i)\}$ and the transition kernel $Q(\mathbf{z}|\mathbf{z}')$ as an $N_z \times N_z$ matrix $\mathbf{Q} = \{q_{i,j} = Q(\mathbf{a}_i|\mathbf{a}_j)\}$. Note that the elements in $\mathbf{p}_n$ always add up to unity, as do the columns of $\mathbf{Q}$. This implies that $\mathbf{Q}$ will have at least one eigenvalue equal to 1. We then have

$$\begin{aligned}\mathbf{p}_n &= \mathbf{Q}\mathbf{p}_{n-1} \\ &= \mathbf{Q}^n\mathbf{p}_0.\end{aligned}\tag{5.9}$$

We say that a distribution $\mathbf{p}$ is *invariant* for a Markov chain when

$$\mathbf{p} = \mathbf{Q}\mathbf{p}.\tag{5.10}$$

In other words, it is a right eigenvector with eigenvalue 1. There exists at least one such eigenvector. A distribution $\mathbf{p}$ is *limiting* for a Markov chain when it is invariant and, for every initial distribution $\mathbf{p}_0$, $\mathbf{p} = \mathbf{Q}^n\mathbf{p}_0$, for some finite n .

Sampling from Markov Chains

Suppose that we have a target distribution $p(\mathbf{z})$ from which we wish to draw samples. We write $p(\mathbf{z})$ in vector notation $\mathbf{p}$. Assume that we can find a transition kernel $\mathbf{Q}$ for which $\mathbf{p}$ is a limiting distribution.

We draw $\mathbf{z}^{(0)}$ from some initial distribution $p_0(\cdot)$, then $\mathbf{z}^{(1)}$ from $Q(\cdot|\mathbf{z}^{(0)})$, then $\mathbf{z}^{(2)}$ from $Q(\cdot|\mathbf{z}^{(1)})$, and so forth. After some time (known as the burn-in time), we will eventually draw samples from the target distribution $\mathbf{p}$. The samples generated in this way are usually correlated. One way to remedy this problem is to draw samples from parallel streams. We will discuss this issue in the sequel.

Gibbs Sampling

Let $\mathbf{z} = (z_0, z_1, \dots, z_{D-1})$ be a D -dimensional random variable. For simplicity, let us focus on $D = 3$. Now, we set the transition kernel to be

$$\begin{aligned} Q(\mathbf{z}|\mathbf{z}') &= Q(z_0, z_1, z_2|z'_0, z'_1, z'_2) \\ &\doteq p(z_0|z'_1, z'_2)p(z_1|z_0, z'_2)p(z_2|z_0, z_1), \end{aligned} \quad (5.11)$$

where the dot equation means that we are assigning the right hand side to the transition kernel Q . Then, we have

$$\begin{aligned} \sum_{\mathbf{z}'} p(\mathbf{z}')Q(\mathbf{z}|\mathbf{z}') &= \sum_{z'_0, z'_1, z'_2} p(z'_0, z'_1, z'_2)p(z_0|z'_1, z'_2)p(z_1|z_0, z'_2)p(z_2|z_0, z_1) \\ &= p(z_0)p(z_1|z_0)p(z_2|z_0, z_1) \\ &= p(z_0, z_1, z_2) \\ &= p(\mathbf{z}). \end{aligned}$$

In other words, $p(\mathbf{z})$ is an invariant distribution for the transition kernel $Q(\mathbf{z}|\mathbf{z}')$. Moreover, it turns out that $p(\mathbf{z})$ is a limiting distribution for the Markov chain. For a given z'_0, z'_1, z'_2 , sampling from the transition kernel is performed as follows:

- draw sample z_0 from $p(z_0|z'_1, z'_2)$,
- draw sample z_1 from $p(z_1|z_0, z'_2)$,
- draw sample z_2 from $p(z_2|z_0, z_1)$.

In other words, we only need to sample from conditional distributions. Compared to the original D -dimensional distribution, these one-dimensional conditional distributions are commonly much easier to sample from.

For the general case of a D -dimensional distribution, the Gibbs sampler operates as follows: given an initial state $\mathbf{z}^{(-B)} = (z_0^{(-B)}, z_1^{(-B)}, \dots, z_{D-1}^{(-B)})$, we draw samples $\mathbf{z}^{(k)} = (z_0^{(k)}, z_1^{(k)}, \dots, z_{D-1}^{(k)})$ for $-B + 1 \leq k \leq N_s$ in the following way:

- draw sample $z_0^{(k)}$ from $p(z_0|z_1^{(k-1)}, z_2^{(k-1)}, \dots, z_{D-1}^{(k-1)})$,
- draw sample $z_1^{(k)}$ from $p(z_1|z_0^{(k)}, z_2^{(k-1)}, \dots, z_{D-1}^{(k-1)})$,
- $\vdots$
- draw sample $z_{D-1}^{(k)}$ from $p(z_{D-1}|z_0^{(k)}, z_1^{(k)}, \dots, z_{D-2}^{(k)})$.

The parameter B is referred to as the burn-in period, as the samples for $-B \leq k \leq 0$, are usually discarded. As mentioned before, the samples generated by the Gibbs sampler are correlated. In order to make the samples less correlated, it is helpful to generate samples using parallel Gibbs samplers. However, this method has the drawback that each sampler requires its own burn-in period.

5.4 MCMC Based 2-D Detection Method

The MCMC based detection method has already been used over 1-D ISI channels [41]. In this work, we propose to apply the MCMC based method to 2-D ISI channels, and to the best of our knowledge, this is the first time that the MCMC based technique is used for 2-D ISI channels.

Referring to the 2-D ISI channel model and the turbo decoding scheme discussed in Section 5.1, we aim to design a detector to compute the LLR

$$L(x_{m,n}|\mathbf{Y}) = \log \left(\frac{Pr(x_{m,n} = -1|\mathbf{Y})}{Pr(x_{m,n} = +1|\mathbf{Y})} \right), \quad (5.12)$$

given the received matrix $\mathbf{Y}$. In this section, we propose to estimate the *a posteriori* probabilities $Pr(x_{m,n}|\mathbf{Y})$ using Monte Carlo techniques. We propose the algorithm both in probability and log domains.

5.4.1 Probability Domain MCMC Based Detector

Calculating the conditional probability $Pr(x_{m,n}|\mathbf{Y})$ involves a multi-dimensional summation operation that is well suited for the MCMC methods. For a given $\mathbf{Y}$, $Pr(x_{m,n}|\mathbf{Y})$ can be expressed as

$$\begin{aligned} Pr(x_{m,n}|\mathbf{Y}) &= \sum_{\mathbf{X}_{\overline{m,n}}} Pr(\mathbf{X}|\mathbf{Y}) \\ &= \sum_{\mathbf{X}_{\overline{m,n}}} Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y}) Pr(\mathbf{X}_{\overline{m,n}}|\mathbf{Y}), \end{aligned} \quad (5.13)$$

where $\mathbf{X}_{\overline{m,n}}$ represents all the symbols in $\mathbf{X}$ excluding $x_{m,n}$. Now, let us set $\mathbf{z} \doteq \mathbf{X}_{\overline{m,n}}$, $h(\mathbf{z}) \doteq Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y})$ and $f(\mathbf{z}) \doteq Pr(\mathbf{X}_{\overline{m,n}}|\mathbf{Y})$. Then, the desired *a posteriori* probability $Pr(x_{m,n}|\mathbf{Y})$ will be equal to the expected value of $h(\mathbf{z})$, i.e., $Pr(x_{m,n}|\mathbf{Y}) = E\{h(\mathbf{z})\} = E\{Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y})\}$. Next, $h(\mathbf{z})$ can be approximated using (5.5).

We first draw N_s samples $\{\mathbf{X}_{\overline{m,n}}^{(k)}\}_{k=1}^{N_s}$ from $Pr(\mathbf{X}_{\overline{m,n}}|\mathbf{Y})$ using a Gibbs sampler. Next, with regard to (5.5), $Pr(x_{m,n}|\mathbf{Y})$ is approximated as

$$Pr(x_{m,n}|\mathbf{Y}) \approx \frac{1}{N_s} \sum_{k=1}^{N_s} Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}^{(k)}, \mathbf{Y}). \quad (5.14)$$

In the following, we describe the detailed computation of different quantities in the above summation, and we explain how we construct the Gibbs sampler.

We first briefly look into the factor $p(\mathbf{Y}|\mathbf{X}_{\overline{m,n}}^{(k)}, x_{m,n})$, which is required in our computations. Using the channel model of (5.3), and arranging factors, we can write

$$p(\mathbf{Y}|\mathbf{X}_{\overline{m,n}}^{(k)}, x_{m,n}) = \alpha_{m,n}^{(k)} \beta_{m,n}^{(k)}(x_{m,n}), \quad (5.15)$$

where

$$\alpha_{m,n}^{(k)} = \prod_{\substack{0 \leq r \leq M+L_M-1, 0 \leq q \leq N+L_N-1 \\ (r,q) \notin \{m:m+L_M, n:n+L_N\}}} p\left(y_{r,q} | g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)}\right)\right) \quad (5.16)$$

and

$$\beta_{m,n}^{(k)}(x_{m,n}) = \prod_{(r,q) \in \{m:m+L_M, n:n+L_N\}} p\left(y_{r,q} | g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \middle| x_{m,n} \right)\right) \quad (5.17)$$

with $\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \middle| x_{m,n}$ signifying that the entry corresponding to index (m, n) in $\mathbf{X}^{(k)}$ is set to $x_{m,n}$. Recalling the observation model (5.3), we have

$$p\left(y_{r,q} | g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \right)\right) \propto \exp\left(-\frac{1}{2\sigma^2} \left(y_{r,q} - g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \right)\right)^2\right). \quad (5.18)$$

Now, we describe the steps of the algorithm.

Sampling from $Pr(\mathbf{X}|\mathbf{Y})$

We use the Gibbs sampler to draw samples from $Pr(\mathbf{X}|\mathbf{Y})$, relying on the conditional distributions $Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y})$. For the k th sample generation, the components of $\mathbf{X}_{\overline{m,n}}$ are obtained from either the k th and/or the $(k-1)$ th sample, as described in the algorithm of the Gibbs sampler in Section 5.3. Note that here $\mathbf{X}$ is a matrix where in Section 5.3, $\mathbf{z}$, the random variable involved in the Gibbs sampler, was considered to be a vector. To deal with this issue, we consider the components of $\mathbf{X}$ as a sequence by sweeping the elements of $\mathbf{X}$ either row by row or column by column. Next, we can express $Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y})$ as

$$\begin{aligned} Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y}) &= \frac{1}{p(\mathbf{Y}|\mathbf{X}_{\overline{m,n}})} p(\mathbf{Y}|\mathbf{X}_{\overline{m,n}}, x_{m,n}) Pr(x_{m,n}) \\ &= \gamma Pr(x_{m,n}) \beta_{m,n}(x_{m,n}), \end{aligned} \quad (5.19)$$

where the last line was obtained from (5.15) by absorbing all factors that do not depend on $x_{m,n}$ in a normalization constant γ . The functions $\alpha_{m,n}$ and $\beta_{m,n}(x_{m,n})$ are defined similar to $\alpha_{m,n}^{(k)}$ and $\beta_{m,n}^{(k)}(x_{m,n})$. For a given $\mathbf{X}_{\overline{m,n}}$ and a given $\mathbf{Y}$, $Pr(x_{m,n})\beta_{m,n}(x_{m,n})$ can be determined for every value of $x_{m,n} \in \{-1, +1\}$. Normalizing then immediately yields γ . Finally, we can now sample from $Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}, \mathbf{Y})$.

Sampling from $Pr(\mathbf{X}_{\overline{m,n}}|\mathbf{Y})$

Given N_s samples $\{\mathbf{X}^{(k)}\}_{k=1}^{N_s}$ from $Pr(\mathbf{X}|\mathbf{Y})$, samples from $Pr(\mathbf{X}_{\overline{m,n}}|\mathbf{Y})$ can be obtained by simply dropping the (m, n) th component in every sample $\mathbf{X}^{(k)}$.

Computation of $Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}^{(k)}, \mathbf{Y})$

Applying (5.19) immediately gives us

$$Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}^{(k)}, \mathbf{Y}) = \gamma^{(k)} Pr(x_{m,n}) \beta_{m,n}^{(k)}(x_{m,n}), \quad (5.20)$$

where $\gamma^{(k)}$ is a normalization constant such that $\sum_{x_{m,n} \in \{-1, +1\}} Pr(x_{m,n}|\mathbf{X}_{\overline{m,n}}^{(k)}, \mathbf{Y}) = 1$.

Computation of $Pr(x_{m,n}|\mathbf{Y})$

Finally, $Pr(x_{m,n}|\mathbf{Y})$ can be approximated using (5.14).

In practice, the algorithm is implemented in log domain to achieve more numerical stability and less complexity.

5.4.2 Log Domain MCMC Based Detector

Referring to (5.17), we replace $\beta_{m,n}^{(k)}(x_{m,n})$ by

$$\begin{aligned} \tilde{\beta}_{m,n}^{(k)}(x_{m,n}) &\triangleq \log(\beta_{m,n}^{(k)}(x_{m,n})) \\ &= \log \left(\prod_{(r,q) \in \{m:m+L_M, n:n+L_N\}} p \left(y_{r,q} | g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \middle| x_{m,n} \right) \right) \right) \\ &= \sum_{(r,q)} \log \left(p \left(y_{r,q} | g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \middle| x_{m,n} \right) \right) \right), \end{aligned} \quad (5.21)$$

and using (5.18), we have

$$\log \left(p \left(y_{r,q} | g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \right) \right) \right) = d - \frac{1}{2\sigma^2} \left(y_{r,q} - g_{r,q} \left(\mathbf{X}_{r-L_M:r, q-L_N:q}^{(k)} \right) \right)^2, \quad (5.22)$$

5.4 MCMC Based 2-D Detection Method

75

where d is a constant number. Next, let $L_{m,n}^{dec}$ be the *prior* information of the (m, n) symbol defined by

$$L_{m,n}^{dec} \triangleq \log \left(\frac{Pr(x_{m,n} = -1)}{Pr(x_{m,n} = +1)} \right), \quad (5.23)$$

which is fed back from the decoder into the detector.

For the Gibbs sampler, we need to draw samples from conditional distributions obtained by (5.19). In order to implement the sampling in log domain, we define δ using (5.19) and (5.23) as

$$\begin{aligned} \delta &\triangleq \log \left(\frac{Pr(x_{m,n} = -1 | \mathbf{X}_{\overline{m,n}}, \mathbf{Y})}{Pr(x_{m,n} = +1 | \mathbf{X}_{\overline{m,n}}, \mathbf{Y})} \right) \\ &= \log \left(\frac{Pr(x_{m,n} = -1)}{Pr(x_{m,n} = +1)} \right) + \log(\beta_{m,n}(x_{m,n} = -1)) - \log(\beta_{m,n}(x_{m,n} = +1)) \\ &= L_{m,n}^{dec} + \tilde{\beta}_{m,n}(x_{m,n} = -1) - \tilde{\beta}_{m,n}(x_{m,n} = +1), \end{aligned} \quad (5.24)$$

where $\tilde{\beta}_{m,n}(x_{m,n})$ is defined in a similar way to (5.21). Note that the constant term of $\tilde{\beta}_{m,n}(x_{m,n})$ is not required to be evaluated, as it cancels out in (5.24).

Next, using (5.19) and (5.24) we can write

$$\begin{aligned} Pr(x_{m,n} = -1 | \mathbf{X}_{\overline{m,n}}, \mathbf{Y}) &= \frac{1}{1 + \frac{Pr(x_{m,n} = +1 | \mathbf{X}_{\overline{m,n}}, \mathbf{Y})}{Pr(x_{m,n} = -1 | \mathbf{X}_{\overline{m,n}}, \mathbf{Y})}} \\ &= \frac{1}{1 + e^{-\delta}}. \end{aligned} \quad (5.25)$$

We note that (5.25) requires evaluation of a function of the form $1/(1 + e^{-\delta})$ which is difficult to be implemented in hardware. It is common in the literature to use look-up tables for such calculations [42]. Here, we approximate this function by a piece-wise linear function as [43]

$$\frac{1}{1 + e^{-\delta}} \approx \begin{cases} \frac{\delta}{2^3} + \frac{1}{2}, & -4 < x < 4 \\ 1, & x \geq 4 \\ 0, & x \leq -4 \end{cases}. \quad (5.26)$$

Substituting (5.24) in (5.26), we finally get

$$Pr(x_{m,n} = -1 | \mathbf{X}_{\overline{m,n}}, \mathbf{Y}) \approx \frac{1}{2} + \frac{L_{m,n}^{dec} + \tilde{\beta}_{m,n}(x_{m,n} = -1) - \tilde{\beta}_{m,n}(x_{m,n} = +1)}{2^3}. \quad (5.27)$$

5.4 MCMC Based 2-D Detection Method

76

The next step after sampling is to compute $L(x_{m,n}|\mathbf{Y})$. Based on (5.14), and recalling the $\max^*(u, v)$ function in (4.18) and its extension to more than two variables in Fact 4.5, $L(x_{m,n}|\mathbf{Y})$ can be approximated as

$$\begin{aligned}
L(x_{m,n}|\mathbf{Y}) &\approx \log \left(\frac{\sum_{k=1}^{N_s} Pr(x_{m,n} = -1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y})}{\sum_{k=1}^{N_s} Pr(x_{m,n} = +1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y})} \right) \\
&= \log \left(\sum_{k=1}^{N_s} Pr(x_{m,n} = -1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y}) \right) \\
&\quad - \log \left(\sum_{k=1}^{N_s} Pr(x_{m,n} = +1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y}) \right) \\
&= \max_{1 \leq k \leq N_s}^* \left(\log(Pr(x_{m,n} = -1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y})) \right) \\
&\quad - \max_{1 \leq k \leq N_s}^* \left(\log(Pr(x_{m,n} = +1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y})) \right). \tag{5.28}
\end{aligned}$$

From (5.20) and (5.23), we have

$$\log(Pr(x_{m,n}|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y})) = d - \frac{1}{2}x_{m,n}L_{m,n}^{dec} + \tilde{\gamma}^{(k)} + \tilde{\beta}_{m,n}^{(k)}(x_{m,n}), \tag{5.29}$$

where d is a constant number, and $\tilde{\gamma}^{(k)}$ is defined as $\tilde{\gamma}^{(k)} \triangleq \log(\gamma^{(k)})$, and is obtained by the following derivation.

$$\begin{aligned}
Pr(x_{m,n} = -1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y}) + Pr(x_{m,n} = +1|\mathbf{X}_{m,n}^{(k)}, \mathbf{Y}) &= 1 \\
\gamma^{(k)} Pr(x_{m,n} = -1)\beta_{m,n}^{(k)}(x_{m,n} = -1) + \gamma^{(k)} Pr(x_{m,n} = +1)\beta_{m,n}^{(k)}(x_{m,n} = +1) &= 1 \\
\log(\gamma^{(k)}) &= -\log(Pr(x_{m,n} = -1)\beta_{m,n}^{(k)}(x_{m,n} = -1) \\
&\quad + Pr(x_{m,n} = +1)\beta_{m,n}^{(k)}(x_{m,n} = +1)) \\
\tilde{\gamma}^{(k)} &= d' - \max^* \left(\frac{1}{2}L_{m,n}^{dec} + \tilde{\beta}_{m,n}^{(k)}(x_{m,n} = -1), -\frac{1}{2}L_{m,n}^{dec} + \tilde{\beta}_{m,n}^{(k)}(x_{m,n} = +1) \right), \tag{5.30}
\end{aligned}$$

where d' is a constant number and can be absorbed in d . However, constant terms are not required to be evaluated, as they cancel out in (5.28).

At the last stage of the algorithm, the output *extrinsic* information of the detector is obtained by

$$L_{m,n}^{det} = L(x_{m,n}|\mathbf{Y}) - L_{m,n}^{dec}, \tag{5.31}$$

Figure 5.3: Block diagram of the MCMC based detector.

and is passed to the decoder.

The block diagram of the MCMC based detector is depicted in Fig. 5.3.

5.4.3 Parallel Sampling

It has been reported in the literature that at high SNR values, the conditional distributions in the Gibbs sampler become degenerate distributions, so that the Gibbs sampler “gets stuck” in a state far from the correct state [41, 44]. Various methods have been investigated in [45] to remedy this problem. One method to deal with this issue is to use multiple Gibbs samplers in parallel. This results in samples that are less correlated, and increases the chance of getting into the correct state. In this work, the Gibbs samplers are initialized randomly except for one of them which is initialized with an estimation obtained by the decision feedback equalizer (DFE) [46]. In this thesis, we use a simple implementation for the DFE. The algorithm starts with the left topmost symbol of the data block. Since all the symbols that affect the received symbol at this position are “-1”, we can determine the noiseless received symbol at this position for both cases of the transmitted symbol (which might be “-1” or “+1”). Then, using (5.22), we decide this symbol to be “-1” or “+1”. Next, we go to the next symbol in the same topmost row of the data block, and we make a hard decision for this symbol in the same way used for the previous symbol. We continue this process until the last symbol of the row, and then, we go to the next row. In summary, the algorithm makes hard decisions based on previous decisions, and continues until the last symbol of the data block.

5.5 Discussion on Computational Complexity

In this section, we compare the computational complexity of the proposed MCMC based detection algorithm with that of the IRCSDFA in terms of the number of required operations per symbol detection. The proposed algorithm can be divided into two steps. At the first stage, N_s number of samples are generated using G_s number of parallel Gibbs samplers. Each sampler discards the first B samples for the burn-in period, and then, generates N_s/G_s samples. Here, the conditional probability of each symbol is obtained by (5.27). At the second stage, $L(x_{m,n}|\mathbf{Y})$ is computed using (5.28), (5.29), and (5.30). The computational burden of this algorithm is mostly dominated by the computation of $\tilde{\beta}_{m,n}(x_{m,n})$ and $\tilde{\beta}_{m,n}^{(k)}(x_{m,n})$. Let S be the ISI size defined by $S \triangleq (L_M + 1)(L_N + 1)$. Then, the computation of $\tilde{\beta}_{m,n}(x_{m,n})$ or $\tilde{\beta}_{m,n}^{(k)}(x_{m,n})$ according to (5.21) involves the summation of S terms obtained by (5.22). Each term, in turn, requires: 1) the evaluation of the noiseless channel output which needs S number of memory access operations (performed once for both $x_{m,n} = -1$ and $x_{m,n} = +1$); 2) the subtraction of the noiseless channel output from the received symbol; and 3) the squaring of the subtraction result. Note that the $1/2\sigma^2$ coefficient in (5.22) can be absorbed inside the squaring operation, and it can be avoided by multiplying its square root by all the received symbols and noiseless channel outputs only once at the beginning of the algorithm. Besides, we consider a look-up table for the noiseless channel outputs. Therefore, the computation of $\tilde{\beta}_{m,n}(x_{m,n})$ or $\tilde{\beta}_{m,n}^{(k)}(x_{m,n})$ for $x_{m,n} = -1$ and $x_{m,n} = +1$ involves $2S$ number of additions, $S \times S$ number of memory access operations, $2S$ number of subtractions, and $2S$ number of multiplications. Assuming that subtractions and memory access operations impose a complexity equal to additions, we end up with $S^2 + 4S$ number of additions and $2S$ number of multiplications. Now, for the first stage, $N_s + G_s B$ samples are generated which results in $(N_s + G_s B)(S^2 + 4S)$ number

5.5 Discussion on Computational Complexity

79

Table 5.1: Computational requirements of the MCMC based algorithm and the IRCSDFA

	MCMC	IRCSDFA
Additions	$(2N_s + G_s B)(S^2 + 4S) + 10N_s$	$I(2^{2S-\sqrt{S}+2})$
Multiplications	$2(2N_s + G_s B)S$	$I(2^{2S-\sqrt{S}+1})$

of additions and $2(N_s + G_s B)S$ number of multiplications per symbol. For the second stage, the N_s samples of the previous stage are used in (5.28). Referring to (5.29) and (5.30), $S^2 + 4S$ number of additions and $2S$ number of multiplications are required for the computation of $\tilde{\beta}_{m,n}^{(k)}(x_{m,n} = \pm 1)$ for each sample. Moreover, (5.29) and (5.30) involve one $\max^*(\cdot)$ operation, one memory access operation for loading the *prior* information, and 6 additions for both $x_{m,n} = -1$ and $x_{m,n} = +1$. Besides, $2N_s$ number of $\max^*(\cdot)$ operations should be performed for the computation of (5.28). Using a look-up table, the complexity of the $\max^*(\cdot)$ function may be assumed to be equal to an addition operation. To sum up, the overall number of additions for the proposed algorithm is $(2N_s + G_s B)(S^2 + 4S) + 10N_s$, and the overall number of multiplications is $2(2N_s + G_s B)S$.

For the IRCSDFA, we set the width of the strips to $L_M + 1$, and we assume that $L_M = L_N$. The number of additions and multiplications for the MCMC based algorithm and the IRCSDFA, are compared in Table 5.1. In this table, I is the number of iterations between the row-by-row and column-by-column detectors for the IRCSDFA. It can be observed from this table that the complexity of the IRCSDFA grows exponentially with the ISI size while the complexity of the MCMC based algorithm grows polynomially.

5.6 Simulation Results

In this section, we investigate the performance of our proposed MCMC based detection algorithm over coded Gaussian 2-D ISI channels. We compare our results with the iterative row-column soft-decision feedback algorithm (IRCSDFA) which has been reported to have better performance than the other proposed sub-optimal detection methods over 2-D ISI channels [34, 35].

Recalling the system model introduced in Section 5.1, we define the signal to noise ratio (SNR) as

$$SNR \triangleq 10 \log_{10} \left(\frac{\sum_{i=0}^{L_M} \sum_{j=0}^{L_N} h_{i,j}^2}{2R_c\sigma^2} \right), \quad (5.32)$$

where $R_c = K/N_c$ is the rate of the LDPC code used. In this section, we use two types of LDPC codes. The first code is a regular Gallager LDPC code of length $N_c = 1998$, rate $R_c = 0.89$, and column weight 4. Each codeword is split into $M = 6$ sequences of length $N = 333$ in order to form the block of transmitted data ($M \times N = N_c = 1998$). The second code is a short length code used for evaluating the performance of the algorithms at low bit error probabilities. It is a type-I EG-LDPC code of length $N_c = 63$ and rate $R_c = 0.59$. For this code, we set $M = 7$ and $N = 9$. For all of our simulations, we use the belief propagation algorithm for decoding the LDPC code, and the maximum number of iterations for the decoder is set to 30. Next, we consider a 2-D ISI channel whose impulse response is given by

$$\mathbf{H} = \begin{bmatrix} a & b & a \\ c & 1 & c \\ a & b & a \end{bmatrix}$$

with $a = 0.050684$, $b = 0.21273$, and $c = 0.23825$. Such channel coefficients correspond to an ultra-high density BPMR system [47]. For our MCMC based algorithms, we use $G_s = 5$ parallel Gibbs samplers, where the burn-in period is set to

$B = 5$, and the number of main samples is set to 10 for each sampler. Therefore, the total number of samples will be $N_s = 50$. The Gibbs samplers are initialized randomly except for one of them which is initialized by DFE. We refer to our proposed algorithm as MCMC-2D. For the IRCSDFA, we consider two schemes: 1) The number of iterations between the row-by-row and column-by-column detectors is set to $I = 3$, and the *extrinsic* information are not scaled ($w = 1$). We refer to this algorithm as IRCSDFA-I. 2) The number of iterations between the row-by-row and column-by-column detectors is set to $I = 6$, and the *extrinsic* information are weighted according to the schedule discussed in [34] ($w(i) = 0.008 \times (3 \times i^2 + 1)$, where i is the iteration number). We refer to this method as IRCSDFA-II. For both of this approaches, we consider 3 rows (or columns) in constructing the trellis of the modified BCJR algorithm as described in [34] (the width of each strip is set to 3). Finally, the number of turbo iterations between the detector and the decoder is set to 5.

Fig. 5.4 shows the simulation results of the IRCSDFA-I, IRCSDFA-II, and MCMC-2D algorithm at the first turbo iteration using the Gallager LDPC code. It can be observed that the IRCSDFA-II outperforms other algorithms, and at the bit error probability of 10^{-3} , it has a coding gain of about 0.3 dB over the MCMC-2D algorithm and about 0.1 dB over the IRCSDFA-I. Although the IRCSDFA-II shows a better BER performance compared to the MCMC based algorithm, we should take note that its computational complexity is much higher than that of the MCMC based algorithm. Furthermore, the IRCSDFA-II is not suitable for the turbo decoding scheme, as according to our simulations, no improvement can be achieved by increasing the number of turbo iterations (the curves have not been depicted as they just fall on top of each other, and the result after the next iteration is essentially the same as the previous result). The reason is that the weighting schedule used for

this algorithm multiplies the LLRs by a very small number close to zero at the first iteration between the row-by-row and column-by-column detectors ($w(1) = 0.032$). Consequently, the input LLRs of the detector at the first iteration fade away inside the detector, and therefore, this algorithm does not incorporate the information obtained from the decoder efficiently. It is very likely to get better performance by employing different set of coefficients, but unfortunately there is no analytical method for determining these coefficients, and brute force search is cumbersome and not practical. Therefore, in the next simulations, we will only compare our results with the IRCSDFA-I which assumes all the coefficients to be one. The IRCSDFA-II with optimal coefficients (if determined by some means) would probably perform better than the other algorithms, but as stated previously, the main advantage of our proposed algorithm is its low complexity while its performance is in the same order as the IRCSDFA.

Next, we investigate the performance improvement achieved by increasing the number of turbo iterations using the same Gallager LDPC code. Fig. 5.5 illustrates the bit error probabilities for all the turbo iterations from 1 to 5. From this figure, we can see how the bit error probability decreases as the number of turbo iterations increases. Furthermore, it turns out that the MCMC-2D algorithm outperforms the IRCSDFA-I after the second turbo iteration. We remind that the computational complexity of the MCMC based algorithm is much less than that of the trellis based algorithms such as the IRCSDFA. In Fig. 5.5, it is observed that at a bit error probability near 10^{-4} and for the 5th turbo iteration, the MCMC-2D algorithm achieves a coding gain of about 0.2 dB over the IRCSDFA-I. The block error probabilities are depicted in Fig. 5.6. From this figure, we can make similar observations as noticed in Fig. 5.5. To sum up, the MCMC-2D algorithm shows a better performance compared to the IRCSDFA, albeit with a lower detection complexity.

5.6 Simulation Results

83

Figure 5.4: Bit and block error probabilities of the IRCSDFA and MCMC based algorithm at the first turbo iteration using the Gallager LDPC code.

In order to compare the performance of our proposed algorithm with the IRCSDFA in the high SNR region which corresponds to low bit error probabilities, we used the second LDPC code which has a shorter length than the first code. The simulation results are depicted in Fig. 5.7. It is observed that similar to our previous simulations, the MCMC-2D algorithm outperforms the IRCSDFA-I.

Finally, we note that the performance of the IRCSDFA can be improved by increasing the width of the strips. However, this imposes a large increase on the computational burden of the algorithm. We remind that the main advantage of our proposed MCMC based detection algorithm is its low complexity. The computational requirements of the MCMC-2D algorithm and the IRCSDFA-I are compared in Table 5.2. From this table, we infer that the number of required additions and multiplications for the MCMC-2D algorithm are 26 and 87 times less than the required additions and multiplications for the IRCSDFA respectively.

Figure 5.5: Bit error probabilities of the IRCSDFA and MCMC based algorithm for five turbo iterations using the Gallager LDPC code.

Table 5.2: Computational complexity comparison of the MCMC-2D algorithm and the IRCSDFA-I

	MCMC-2D	IRCSDFA-I
Additions	15125	393216
Multiplications	2250	196608

5.6 Simulation Results

85

Figure 5.6: Block error probabilities of the IRCSDFA and MCMC based algorithms for five turbo iterations using the Gallager LDPC code.

Figure 5.7: Bit and block error probabilities of the IRCSDFA and MCMC based algorithms at the 5th turbo iteration using the EG-LDPC code.

Chapter 6

Conclusion and Recommendations

6.1 Conclusion

In this thesis, we investigated low-complexity detection and decoding algorithms for low-density parity-check (LDPC) codes applied over data storage channels with intersymbol interference (ISI). Our main objective was to improve the performance of the sub-optimal algorithms or propose methods with lower computational complexity.

We used LDPC codes based on finite geometries because of their good performance for finite-length LDPC codes and their structure which grants a simple encoder and decoder implementation.

LDPC codes can be decoded by various decoding algorithms including the belief propagation (BP) algorithm, the min-sum (MS), normalized MS (NMS), and offset MS (OMS) algorithms. We proposed the combined normalized and offset (NOMS) algorithm as a generalization/combination of the NMS and OMS algorithms. In Chapter 3, we investigated the performance of the NOMS algorithm over AWGN and Rayleigh fading channels, and in Chapter 4, we presented its performance over partial response channels. It was shown that this algorithm outperforms the NMS

and OMS algorithms by imposing a slight increase in complexity.

In Chapter 5, we considered the channel detection problem over two-dimensional (2-D) ISI channels. Instances of these channels are 2-D optical recording and bit-patterned magnetic recording (BPMR) systems in which both the conventional down-track ISI and cross-track ISI exist to result in 2-D ISI. In contrast to 1-D channels, where optimal detection is applicable using the BCJR and Viterbi algorithms, there does not exist a practical implementation of an optimal detection method for 2-D channels because of the prohibitive computational complexity.

There are a number of sub-optimal algorithms for channel detection over 2-D ISI channels. However, the computational complexity of these algorithms is still too high for general applications. In Chapter 5, we proposed a reduced-complexity algorithm for detection over 2-D ISI channels using Markov chain Monte Carlo (MCMC) techniques. We showed that not only the computational complexity of our proposed algorithm is much less than that of the existing sub-optimal algorithms, but also the complexity of our algorithm grows polynomially with the ISI size, instead of exponentially. Moreover, we showed that our proposed algorithm even outperforms the method based on the BCJR algorithm over a 2-D ISI channel whose coefficients correspond to a BPMR system. Note that the BCJR based algorithm is performed for a limited number of rows on each block of data, and therefore it is sub-optimal. Thus, it is possible to get a better performance than this algorithm.

6.2 Recommendations

Our suggestion for future work are as follows:

- 1) In our study we considered linear systems with additive white Gaussian noise. However, the channel model for storage systems may be more complicated. One of

our suggestions for future work is to consider more realistic channel models. As a case in point, in BPMR systems, individual bits are written on isolated magnetic islands to achieve increased thermal stability, and hence increased areal density. However, lithography noise will result in jitter noise which cannot be modeled to be additive. Furthermore, we assumed the noise to be white, i.e. we considered the noise samples to be independent from each other. However, this might not be a realistic assumption, and the noise samples may be correlated or even data dependent. Our suggestion for future work is to first study practical channel models, and then modify our proposed algorithms to remedy the possible problems caused by the channel impairments.

2) In our MCMC based detection algorithms, we used parallel Gibbs samplers to draw samples from a target distribution. In our simulations, we found that initialization of the Gibbs samplers has a great impact on the performance of the algorithms. In this thesis, we initialized the samplers using a uniform random distribution. Besides, we initialized one of the samplers using the decision feedback equalizer. Our suggestion for future work is to investigate other methods for initializing samplers. For example, prior information may be deployed to derive a non-uniform distribution used in the initialization step.

List of Publications

- [1] M. Shaghaghi, Y. L. Guan, K. Cai, and Z. Qin, “Combined normalized and offset min-sum decoding over partial response channels,” ICICS, Dec. 2009.
- [2] M. Shaghaghi, K. Cai, Y. L. Guan, and Z. Qin, “Monte Carlo based detection for two-dimensional intersymbol interference channels,” submitted to *IEEE Trans. Magn.*

Bibliography

- [1] R. G. Gallager, *Low Density Parity Check Codes*. MA: MIT Press, 1963.
- [2] D. J. C. MacKay and R. M. Neal, “Near Shannon limit performance of low density parity check codes,” *Electronics Letters*, vol. 32, no. 18, p. 1645, Aug. 1996.
- [3] ETSI, “Digital video broadcasting (DVB); second generation framing structure, channel coding and modulation systems for broadcasting, interactive services, news gathering and other broadband satellite applications (DVB-S2),” EN 302 307, 2004.
- [4] *Part 16: Air Interface for Fixed and Mobile Broadband Wireless Access Systems Amendment for Physical and Medium Access Control Layers for Combined Fixed and Mobile Operation in Licensed Bands*. IEEE P802.16e-2005, 2005.
- [5] J. G. Proakis and M. Salehi, *Communication Systems Engineering*. Upper Saddle River, New Jersey: Prentice-Hall, 2002.
- [6] Y. Kou, S. Lin, and M. Fossorier, “Low-density parity-check codes based on finite geometries: A rediscovery and new results,” *IEEE Trans. Inform. Theory*, vol. 47, no. 7, pp. 2711–2736, Nov. 2001.

-
- [7] T. Richardson, A. Shokrollahi, and R. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619–637, Feb. 2001.
 - [8] Z. Li, L. Chen, L. Zeng, S. Lin, and W. Fong, "Efficient encoding of quasi-cyclic low-density parity-check codes," *IEEE Trans. Commun.*, vol. 54, no. 1, pp. 71–81, Jan. 2006.
 - [9] Z. Wang and Z. Cui, "Low-complexity high-speed decoder design for quasi-cyclic LDPC codes," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 1, pp. 104–114, Jan. 2007.
 - [10] T. Richardson and R. Urbanke, *Modern Coding Theory*. New York: Cambridge University Press, 2008.
 - [11] S.-Y. Chung, J. G. D. Forney, 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, no. 2, pp. 58–60, Feb. 2001.
 - [12] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. Upper Saddle River, New Jersey: Prentice-Hall, 2004.
 - [13] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Mateo, CA: Morgan Kaufmann, 1988.
 - [14] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inform. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.
 - [15] T. Richardson and R. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

-
- [16] H. Jilei, P. H. Siegel, and L. B. Milstein, "Performance analysis and code optimization of low density parity-check codes on Rayleigh fading channels," *IEEE J. Select. Areas Commun.*, vol. 19, no. 5, pp. 924–934, May 2001.
 - [17] M. Fossorier, M. Mihaljevic, and H. Imai, "Reduced complexity iterative decoding of low-density parity check codes based on belief propagation," *IEEE Trans. Commun.*, vol. 47, no. 5, pp. 673–680, May 1999.
 - [18] J. Chen and M. P. C. Fossorier, "Near-optimum universal belief-propagation-based decoding of low-density parity-check codes," *IEEE Trans. Commun.*, vol. 50, no. 3, pp. 406–414, Mar. 2002.
 - [19] —, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.
 - [20] J. Zhang, M. Fossorier, D. Gu, and J. Zhang, "Two-dimensional correction for min-sum decoding of irregular LDPC codes," *IEEE Commun. Lett.*, vol. 10, pp. 180–182, Mar. 2006.
 - [21] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.
 - [22] T. Nishiyama, K. Tsukano, T. Hirai, T. Nara, and S. Mita, "An EEPR4 channel with an error-correcting post-processor designed for 16/17 rate quasi-MTR code," *IEEE GlobComm*, pp. 2868–2873, Nov. 1998.
 - [23] H. Song, R. M. Todd, and J. R. Cruz, "Low density parity check codes for magnetic recording channels," *IEEE Trans. Magn.*, vol. 36, no. 5, pp. 2183–2186, Sept. 2000.

- [24] B. Vasic and E. M. Kurtas, *Coding and Signal Processing for Magnetic Recording Systems*. Boca Raton, FL: CRC, 2005.
- [25] E. Ordentlich and R. M. Roth, "On the computational complexity of 2D maximum-likelihood sequence detection," Hewlett-Packard Laboratories, Palo Alto, CA, Tech. Rep. HPL-2006-69, Apr. 2006. [Online]. Available: <http://www.hpl.hp.com>.
- [26] K. M. Chugg, X. Chen, and M. A. Neifeld, "Two-dimensional linear MMSE for page-oriented optical memories," *31st Annual Asilomar Conf. Sig., Sys. and Comps.*, vol. 1, pp. 343–347, Nov. 1997.
- [27] ———, "Two-dimensional equalization in coherent and incoherent page-oriented optical memory," *J. Opt. Soc. Amer. A*, vol. 16, no. 3, pp. 549–562, Mar. 1999.
- [28] N. Singla, J. A. O'Sullivan, R. S. Indeck, and Y. Wu, "Iterative decoding and equalization for 2-D recording channels," *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2328–2330, Sept. 2002.
- [29] Y. Wu and J. A. O'Sullivan, "Iterative detection and decoding for separable two-dimensional intersymbol interference," *IEEE Trans. Magn.*, vol. 39, no. 4, pp. 2115–2120, Jul. 2003.
- [30] A. H. J. Immink, W. M. J. Coene, A. M. van der Lee, C. Busch, A. P. Hekstra, J. W. M. Bergmans, J. Riani, S. J. L. V. Beneden, and T. Conway, "Signal processing and coding for two-dimensional optical storage," *IEEE GlobComm*, pp. 3904–3908, Dec. 2003.
- [31] M. Marrow and J. K. Wolf, "Iterative detection of 2-dimensional ISI channels," in *Proc. IEEE Information Theory Workshop*, Paris, France, Mar./Apr. 2003, pp. 131–134.

-
- [32] O. Shental, N. Shental, A. J. Weiss, and Y. Weiss, “Generalized belief propagation receiver for near-optimal detection of two-dimensional channels with memory,” in *Proc. IEEE Information Theory Workshop*, San Antonio, TX, Oct. 2004.
 - [33] O. Shental, N. Shental, S. Shamai (Shitz), I. Kanter, A. J. Weiss, and Y. Weiss, “Discrete-input two-dimensional Gaussian channels with memory: estimation and information rates via graphical models and statistical mechanics,” *IEEE Trans. Inform. Theory*, vol. 54, no. 2, pp. 1500–1513, Apr. 2008.
 - [34] T. Cheng, B. J. Belzer, and K. Sivakumar, “Row-column soft-decision feedback algorithm for two-dimensional intersymbol interference,” *IEEE Signal Processing Lett.*, vol. 14, no. 7, pp. 433–436, Jul. 2007.
 - [35] Y. Zhu, T. Cheng, K. Sivakumar, and B. J. Belzer, “Markov random field detection on two-dimensional intersymbol interference channels,” *IEEE Trans. Signal Processing*, vol. 56, no. 7, pp. 2639–2648, Jul. 2008.
 - [36] J. S. Yedidia, W. T. Freeman, and Y. Weiss, “Constructing free-energy approximations and generalized belief propagation algorithms,” *IEEE Trans. Inform. Theory*, vol. 51, no. 7, pp. 2282–2312, Jul. 2005.
 - [37] C. P. Robert and G. Casella, *Monte Carlo Statistical Methods*. New York, NY: Springer Verlag, 1999.
 - [38] D. J. C. MacKay, “Introduction to Monte Carlo methods,” in *Learning in Graphical Models*, ser. NATO Science Series, M. I. Jordan, Ed. Kluwer Academic Press, 1998, pp. 175–204.
 - [39] H. Wymeersch, *Iterative Receiver Design*. Cambridge University Press, 2007.

-
- [40] W. R. Gilks, S. Richardson, and D. J. Spiegelhalter, *Markov Chain Monte Carlo in Practice*. Chapman & Hall, 1995.
 - [41] F. M. Kashif, H. Wymeersch, and M. Z. Win, "Monte Carlo equalization for nonlinear dispersive satellite channels," *IEEE J. Select. Areas Commun.*, vol. 26, no. 2, pp. 245–255, Feb. 2008.
 - [42] P. Robertson, E. Villebrun, and P. Hoeher, "A comparison of optimal and sub-optimal MAP decoding algorithms operating in the log domain," in *Proc. IEEE ICC*, Seattle, WA, Jun. 1995, pp. 1009–1013.
 - [43] S. A. Laraway and B. Farhang-Boroujeny, "Implementation of a Markov chain Monte Carlo based multiuser/MIMO detector," *IEEE Trans. Circuits Syst.*, vol. 56, no. 1, pp. 246–255, Jan. 2009.
 - [44] H. Zhu, B. Farhang-Boroujeny, and R.-R. Chen, "On performance of sphere decoding and Markov chain Monte Carlo detection methods," *IEEE Signal Processing Lett.*, vol. 12, no. 10, pp. 669–672, Oct. 2005.
 - [45] X. Mao, P. Amini, and B. Farhang-Boroujeny, "Markov chain Monte Carlo MIMO detection methods for high signal-to-noise ratio regimes," *IEEE Glob-Comm*, pp. 3979–3983, Nov. 2007.
 - [46] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication.*, 3rd ed. New York, NY: Springer, 2004.
 - [47] Y. Ng, K. Cai, B. V. K. V. Kumar, S. Zhang, and T. C. Chong, "Channel modeling and two-dimensional equalization for bit-patterned media," in *Intermag 2009*. Digest of Technical Papers, May 2009, pp. 71–72.