Detection and decoding for two-dimensional interference channels

Yao, Jun

2013

Yao, J. (2013). Detection and decoding for two-dimensional interference channels. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/55101>

<https://doi.org/10.32657/10356/55101>

Detection and Decoding for Two-Dimensional Interference Channels

YAO Jun

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2013

Acknowledgement

First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Teh Kah Chan, and my co-supervisor, Dr. Li Kwok Hung, for their professional guidance, enthusiastic assistance, kind encouragement, and unconditional support throughout my four years of PhD study. Without their numerous help, I would not have completed this research work.

Secondly, I would like to thank Dr. Cai Kui and Dr. Qin Zhiliang from Data Storage Institute, A*STAR, and the technicians and fellow students in Communication Lab III for their provided useful materials and invaluable discussions.

Thirdly, I am grateful to Nanyang Technological University for providing me the research scholarship and the study facilities.

Finally, I would like to dedicate this thesis to my family. Their endless love is the warmest power to keep me going.

Contents

Acknowledgement	i
Contents	ii
Abstract	vi
List of Abbreviations	viii
List of Figures	xi
List of Tables	xiv
1 Introduction	1
1.1 Background	1
1.2 Motivation	6
1.3 Objectives	6
1.4 Major Contribution	7
1.5 Organization	8
2 Background Knowledge and Literature Review	10
2.1 Message-Passing Algorithm	10
2.1.1 Computing Marginal Functions	12

2.1.2	Sum-Product Algorithm	13
2.2	Low-Density Parity-Check Codes	16
2.2.1	LDPC Encoding	16
2.2.2	LDPC Decoding	17
2.3	LDPC Codes Over 1-D ISI Channels	21
2.4	Detection and Equalization for 2-D ISI Channels	24
2.4.1	Optimal Detection	26
2.4.2	Suboptimal Equalization and Detection	28
3	Maximum-Likelihood Page Detection	33
3.1	Channel Model and MLPD	34
3.2	Probability of Error	36
3.2.1	Independent Error Event	36
3.2.2	Probability of a Particular Independent Error Event	36
3.2.3	Probability of Symbol Error	39
3.3	Numerical Results and Discussions	42
3.4	Conclusion	48
4	Joint Iterative Detection/Decoding Scheme	50
4.1	System Model	51
4.2	Detection/Decoding Algorithm	53
4.2.1	Down-Track Detector	54
4.2.2	Cross-Track Detector	58
4.2.3	Extrinsic Information Transfer	63
4.2.4	Reduced-Complexity Detection	67
4.2.5	Summary of the JIDDS	71
4.3	Simulation Results and Discussions	72

4.3.1	Definitions and Parameters	73
4.3.2	Effect of Parameters on System Performance	74
4.3.3	Performance and Complexity Comparisons	77
4.4	Conclusion	84
5	Joint Message-Passing Decoding Algorithm	85
5.1	Joint Decoding Algorithm	86
5.1.1	State-Processing Node Update	86
5.1.2	Variable-Check Node Update	92
5.1.3	Summary of the JMPDA	94
5.2	EXIT Chart of the Iterative Decoding System	96
5.2.1	Validation of Gaussian Approximation	96
5.2.2	Computation of the Mutual Information	99
5.2.3	EXIT Chart Generating Algorithm	101
5.3	Results and Discussions	104
5.3.1	Effects of Parameters on the EXIT Curves	104
5.3.2	Performance Comparisons	107
5.4	Conclusion	110
6	Performance Analysis of LDPC Codes over 2-D ISI Channels via Density Evolution	112
6.1	System Model	113
6.2	Concentration, Density Evolution and Threshold Calculation	115
6.2.1	Message-Flow Neighborhoods	116
6.2.2	Concentration Theorems	119
6.2.3	Density Evolution and Threshold Calculation	122
6.3	Numerical Results and Discussions	125

6.4 Conclusion	131
7 Conclusions and Future Work	133
7.1 Concluding Remarks	133
7.2 Future Work	135
Author's Publications	140
Bibliography	141
Appendices	155
A Proof of Proposition 6.1	156
B Proof of Theorem 6.1	158
C Proof of Theorem 6.2	160

Abstract

In this thesis, we study the problem of communication over two-dimensional (2-D) interference channels which represent an important class of data storage and communication systems. The maximum-likelihood page detection (MLPD) for 2-D interference channels is firstly studied and the lower and upper performance bounds are derived. Although MLPD is computationally prohibitive and unachievable in practice, the performance bound of MLPD provides a benchmark of the optimal detection of 2-D interference channels. We also propose an approximated bounding method which can accurately predict the performance bound while greatly reducing the computational complexity. A joint iterative detection/decoding scheme (JIDDS) is proposed for 2-D interference system with low-density parity-check (LDPC) codes. The channel detector is composed of two constituent detectors that both adopt trellis-based maximum *a posteriori* probability (MAP) algorithms and they work in the down-track direction and cross-track direction, respectively. To achieve better performance, iterations are introduced between the two constituent detectors and among the constituent detectors with LDPC decoder. Simulation results show that the proposed scheme outperforms some existing algorithms. The better performance of JIDDS is achieved at the cost of higher computational complexity. A reduced-state channel detector is also proposed to save considerable complexity of the system but only degrade system performance marginally.

In JIDDS, the channel detector and LDPC decoder work individually, which may result in some delays in detection/decoding. An appealing practical aspect of the decoding of LDPC codes is that it consists of many small, independent decoding functions (i.e., the variable nodes and check nodes) which can work in parallel. In order to make use of the low-delay advantage of LDPC codes, a joint message-passing decoding al-

gorithm (JMPDA) of LDPC codes and 2-D interference channels is proposed. The channel detector consists of many state-processing nodes which can be implemented in a completely parallel manner in conjunction with the decoding of LDPC codes. The performance of the JMPDA is almost the same as that of the JIDDS. An advantage of the JMPDA is its parallel implementation, leading to a high-speed decoder.

Lastly, the extrinsic information transfer (EXIT) chart and density evolution (DE) are adopted in the analysis of the decoding of LDPC coded 2-D interference systems. The EXIT chart visualizes the evolution of mutual information between channel detection and LDPC decoding during the iterations. The effects of parameters to the decoding system are investigated by analyzing the EXIT curves. It is shown that given a good combination of parameters, the proposed algorithm will converge to the correct result. A graph representation of the JMPDA is explained. It is proved that the message-flow neighborhood of the JMPDA will be tree-like for a sufficiently long code length. It is further proved that the average performance of the JMPDA will concentrate around the performance in which message-flow neighborhood is tree-like. Based on the tree-like message-flow neighborhood, a modified DE algorithm is employed to track the message densities during the iterations. A threshold is then calculated using the DE algorithm which represents the performance limit of the system.

List of Abbreviations

1-D	One-dimensional
2-D	Two-dimensional
ISI	Inter-symbol interference
ITI	Inter-track interference
MLPD	Maximum-likelihood page detection
JIDDS	Joint iterative detection/decoding scheme
LDPC	Low-density parity-check
MAP	maximum <i>a posteriori</i> probability
JMPDA	Joint message-passing decoding algorithm
EXIT	Extrinsic information transfer
DE	Density evolution
HDD	Hard disk drive
MR	Magnetoresistive

ECC	Error-correcting code
VGA	Variable gain amplifier
CTF	Continuous time filter
FIR	Finite impulse response
SNR	Signal-to-noise ratio
PMS	Patterned media storage
HAMR	Heat-assisted magnetic recording
TDMR	Two-dimensional magnetic recording
MIMO	Multiple-input multiple-output
OFDM	Orthogonal-frequency division-multiplexing
APP	<i>A posteriori</i> probability
AWGN	Additive white Gaussian noise
VA	Viterbi algorithm
BCJR	Bahl-Cocke-Jelinek-Raviv
PR	Partial-response
SISO	Soft-input soft-output
SOVA	Soft-output Viterbi algorithm
LLR	Log likelihood ratio

SIR	Symmetric information rate
ML	Maximum likelihood
GPR	Generalized partial response
SSWA	Suboptimal strip-wise algorithm
IRCSDFA	Iterative row-column soft-decision feedback algorithm
MMSE	Minimum mean-squared-error
MLSD	Maximum-likelihood sequence detection
MCMC	Markov chain Monte Carlo
PIE	Partial ITI estimation
BER	Bit-error rate
IDDA	Iterative detection and decoding algorithm
i.i.d.	Independent and identically distributed
i.u.d.	Independent and uniformly distributed
TPC	Turbo product code

List of Figures

1.1	The process of reading signal from magnetic media.	2
1.2	Signals of ideal magnetization transactions.	2
1.3	Signals of real magnetization transactions with interference.	3
1.4	Block diagram of the read channel.	3
1.5	Top view of configuration of the magnetoresistive head and the patterned medium.	5
2.1	Factor graph for the product $f_A(x_1)f_B(x_2)f_C(x_1, x_2, x_3)f_D(x_3, x_4)f_E(x_3, x_5)$	11
2.2	Tanner graph for a typical LDPC code.	19
2.3	Block diagram of LDPC coded ISI channel with turbo equalization.	22
2.4	Markov process for 1-D ISI channel.	27
2.5	2-D Markov random field with bounding strip B making regions A and C conditionally independent.	28
2.6	Optimal 2-D detection by using 1-D detection algorithm over a larger alphabet.	28
2.7	Iterative detection and decoding for separable 2-D ISI channels.	31
3.1	An example of three independent error events.	37
3.2	The upper and lower bounds of the MLPD BER performance.	44
3.3	Performance bound as a function of a and b at SNR=16 dB.	45

3.4	Parts of the spectrums of $d/2\sigma$ for various a and b at SNR=16 dB. . . .	47
3.5	Ratio of the approximated upper bound and the upper bound for various a and b at SNR=16 dB.	48
4.1	Read channel model for a 2-D ISI system.	51
4.2	Factor graph representation of the system.	53
4.3	System diagram of detector and decoder of the proposed scheme. . . .	55
4.4	Block diagram of the iterative channel detector.	55
4.5	Trellis structure of the down-track detector for $M_h = N_h = 3$	56
4.6	Trellis structure of the cross-track detector for $M_h = 3$ and $N_r = 5$. . .	59
4.7	Trellis structure of the reduced-state down-track detector for $N_h = 3$ and $Q = 2$	68
4.8	Effect of the size of the 2-D data array on the system performance. . .	75
4.9	Effect of the interference level of the 2-D ISI channel on the system performance at SNR=10 dB.	76
4.10	Effect of iterations on the system performance for LDPC code (1998,1776). .	78
4.11	BER performance comparisons of uncoded system for channel $\mathbf{H}_A$	79
4.12	BER performance comparisons of uncoded system for channel $\mathbf{H}_B$	80
4.13	BER performance comparisons of JIDDS and MCMC-based algorithm. . .	81
4.14	BER performance comparisons of JIDDS and PIE-based algorithm. . .	82
5.1	Message-passing diagram of a state-processing node.	87
5.2	Message-flow diagram of the joint message-passing decoding algorithm. .	91
5.3	Probability density function $f(\phi)$ of the normalized values of $L_{vs}^3 \cdot x$ and $L_{sv}^3 \cdot x$	97
5.4	Cumulative density function $F(\phi)$ of the normalized values of $L_{vs}^3 \cdot x$ and $L_{sv}^3 \cdot x$	98

5.5	EXIT chart and trajectory of the joint message-passing decoding system.	103
5.6	Effects of T_s on the EXIT curve of state-processing nodes.	105
5.7	Effects of SNR on the EXIT curve of state-processing nodes.	106
5.8	Effects of channel response matrix on the EXIT curve of state-processing nodes.	107
5.9	Effects of parameters on the EXIT curve of variable-check nodes. . . .	108
5.10	Simulation results for different combinations of number of iterations. . .	109
5.11	Performance comparisons of the proposed JMPDA with that of the JIDDS and MCMC-based algorithm.	110
6.1	Message-flow of $L_{sv}^0(x_{i,j})$ and its equivalent representation with $T_s = 1$. .	116
6.2	Message-flow neighborhood of a variable node to a check node and its equivalent representation with $k = 1$ and $T_c = 1$	117
6.3	Message-flow neighborhood of depth k	119
6.4	Simulation results and thresholds for turbo-equalized and non-turbo equalized JMPDA with different LDPC codes and different numbers of iterations.	128
6.5	Simulation results and thresholds for the LDPC codes with $d_v = 3$, $d_c = 6$ and $R = 0.5$	129
6.6	Simulation results and thresholds for the LDPC codes with $d_v = 4$, $d_c = 36$ and $R = 0.89$	130
7.1	1-D to 2-D mapping scheme that results in short cycles	136
7.2	Block diagram of the experimental system.	137
7.3	Performance comparison of systems with different N_w	138

List of Tables

3.1	Six smallest values of $d/2\sigma$ and the corresponding error patterns for various a and b at SNR=16 dB.	46
4.1	Comparison of computational complexity per bit of each individual block	69
5.1	LLR values during the k -th iteration	99
6.1	Evolution of $p_{\text{i.u.d.}}^k$ during the iterations for various parameters.	126
6.2	Threshold values for regular LDPC codes over AWGN and 2-D ISI channels.	126

Chapter 1

Introduction

1.1 Background

Since IBM introduced the Random Access Method of Accounting and Control (RAMAC) system in 1957, the first commercially available disk drive, the hard disk drive (HDD) has been playing a more and more important role in information storage [1]. Due to their nonvolatile, large-capacity and inexpensive features, HDDs become one of the most popular media in data storage nowadays. The pursue of larger capacity HDDs has never stopped. During the past few decades, the recording density of HDDs has increased over 50% annually, which was mainly driven by new technologies. Some of these technologies were related to the development of the hardwares and some others were attributed to new coding and signal processing technologies [2]. For example, the switch from readback heads to magnetoresistive (MR) heads makes the output signal larger and independent on velocity, which is ideal for smaller disks with lower linear velocities. The switch from peak detection to partial response maximum likelihood (PRML) strategy allows a larger degree of inter-symbol interference (ISI). The introduction of some new error-correcting codes (ECCs) [3] provides more robustness to the

Figure 1.1. The process of reading signal from magnetic media.

Figure 1.2. Signals of ideal magnetization transactions.

system and further increased the recording density.

An HDD records data by magnetizing a thin film of ferromagnetic grains on a disk. Changes in the direction of magnetization represent binary data bits. The data is read from the disk by detecting the transitions in magnetization. This process is illustrated in Fig. 1.1. The closer the transitions can be written on the media the higher recording density can be achieved, and therefore, the higher amount of data can be stored. The signals of ideal transitions look like spikes. Spikes or very narrow pulses are less likely to interfere with each other as shown in Fig. 1.2. However, due to characteristics of the read head and the media, the signals of transitions are pulses. As the recording

Figure 1.3. Signals of real magnetization transactions with interference.

Figure 1.4. Block diagram of the read channel.

density becomes higher, adjacent pulses will overlap with each other which cause the interference as shown in Fig. 1.3. The signal from read head is first processed by the front end and then followed by the detector and decoder. The front end processes the analog signal and mainly consists of variable gain amplifier (VGA), continuous time filter (CTF) shaper, sampler and finite impulse response (FIR) filter, etc. The detector and decoder are used to mitigate the interference and decode the ECC, respectively, and there are usually iterations between them. The block diagram of the read channel is depicted in Fig. 1.4. In this thesis, we focus on the detector and decoder of the

read channel. In HDDs, various types of overhead are added to the user data, such as the parity-check bits for ECC. The capacity available to user is equal to the raw capacity minus the capacity used by the overhead. The increase in raw capacity is mostly achieved by increasing the recording density. Thus, it is desirable to use as little space as possible to store the overhead in HDDs. If the overhead takes too much space, in order to maintain a certain capacity available to user, the recording density has to be increased, which will likely increase the interference level and hence degrade the system performance. Therefore, overhead (or equivalent code rate) requirement in HDDs is very stringent.

As the recording density grows, the magnetic grains must also shrink in order to maintain the signal-to-noise ratio (SNR). However, the size of magnetic grains cannot be reduced indefinitely due to some fundamental limitations. Once the sizes of the magnetic grains reach a certain point, the magnetic grains will be magnetically unstable and the recorded data will spontaneously erase after a short period of time. This is called superparamagnetic problem which places a limitation on the recording density of the conventional magnetic recording systems. It has been estimated that the recording density limit of older longitudinal recording HDD is 100 to 200 Gbit/in² [4]. Current hard disk technology uses perpendicular recording. As of August 2010 drives with densities of 667 Gbit/in² are commercially available. Perpendicular recording is predicted to allow recording densities of up to around 1 Tbit/in² (1024 Gbit/in²) [5]. The existing systems are expected to reach their storage density limits in the near future. A lot of research has been done to overcome the superparamagnetic problem to further increase the recording density. Some alternative technologies have been proposed as the potential candidates for the next-generation ultra-high-density data storage systems, such as the patterned media storage (PMS) [6], heat-assisted magnetic recording (HAMR) [7], and two-dimensional magnetic recording (TDMR) [8]. In these systems, the separation

Figure 1.5. Top view of configuration of the magnetoresistive head and the patterned medium.

between tracks is so small that inter-track interference (ITI) cannot be ignored. Along with the conventional ISI, these systems suffer from two-dimensional (2-D) interference. Figure 1.5 depicts the configuration of PMS in which bits are stored in small magnetic islands represented by white grids. The MR head represented by the dark grid is placed on the top of a magnetic island and it senses not only the signal of the desired bit but also signals from the adjacent bits. The ISI and ITI exist in the down-track and cross-track direction, respectively. We refer these 2-D interferences collectively as 2-D ISI. It has been shown that the 2-D ISI degrades the system error performance seriously [9]. 2-D ISI also exists in communication systems. The close relationship between the multiple-input multiple-output (MIMO) communication systems and the 2-D ISI in magnetic recording systems has been studied in [10]. The problem of 2-D ISI also arises with filter bank multi-carrier (FBMC) which was proposed as an alternative approach to multi-carrier orthogonal-frequency division-multiplexing (OFDM) [11].

1.2 Motivation

The problem of 2-D ISI is considered as one of the biggest challenges for the next-generation ultra-high-density data storage systems. The optimal detection for 2-D ISI channels is too complicated to be implemented. Due to different features, the methods used for one-dimensional (1-D) ISI channel detection cannot be directly extended to 2-D ISI channels. Many methods have been proposed to improve the performance of 2-D ISI channel detection. However, there is still considerable performance gap between these suboptimal methods and the optimal detector. Besides, the properties of 2-D ISI channels have not been extensively studied. This motivates us to investigate the 2-D ISI channels and explore methods to further improve the performance.

On the other hand, even if the 2-D ISI can be mitigated effectively, the system performance is still not good enough to satisfy practical applications. In order to improve the system performance, both ECC and joint iterative detection/decoding method need to be applied. Low-density parity-check (LDPC) codes have been shown to possess the capabilities of approaching the Shannon limit and have the potentials to be adopted in magnetic recording systems [12]. Again, this motivates us to investigate the application of LDPC codes to 2-D ISI channels.

1.3 Objectives

The objective of this thesis is to investigate the detection and decoding for 2-D ISI channels. We aim to propose some methods to effectively mitigate the 2-D ISI and jointly implement the detection of 2-D ISI channel and decoding of LDPC codes. We also aim to analyze the performance limit of the LDPC coded 2-D ISI system and investigate the effects of parameters on the system performance. Apart from error

performance of the system, the computational complexity of the system is also a concern throughout this thesis.

1.4 Major Contribution

In this thesis, we first derived the performance bounds of maximum-likelihood page detection (MLPD) over 2-D ISI channels. These performance bounds can be used to assess the performance of the detectors for 2-D ISI channels. Furthermore, with these performance bounds we can analyze the effects of the parameters of the 2-D ISI channels on the system performance. An approximated bounding method has also been proposed to simplify the computation of the bounds. Next, we proposed a joint iterative detection/decoding scheme (JIDDS) for 2-D ISI channels. In this JIDDS, the channel detector consists of two constituent parts which work iteratively in two dimensions. The LDPC codes have been incorporated to further improve the system performance. Monte Carlo simulations were used to demonstrate the superior performance of the JIDDS. It has been shown that the proposed JIDDS outperforms the benchmark which claims to have the best performance results so far. We also analyzed the computational complexity of the JIDDS and proposed a reduced-complexity version of the JIDDS which greatly reduced the system complexity at the expense of very marginal performance loss. We further modified the message-passing update algorithm in JIDDS and proposed a joint message-passing decoding algorithm (JMPDA). This JMPDA is a parallel-implementation version of the JIDDS and provides nearly the same performance as JIDDS but a higher decoding speed. We then used extrinsic information transfer (EXIT) chart technique to analyze the convergence and performance of the proposed iterative detection/decoding system. Moreover, we proved the tree-like structure of the JMPDA and modified the density evolution algorithm to calculate the

thresholds of the JMPDA. The thresholds can be used to fast and accurately predict the performance of the JMPDA with long code length. Finally, we proposed some future research topics.

1.5 Organization

This thesis is divided into 7 chapters. In Chapter 1, a brief introduction on magnetic recording systems is given. The motivation and objective of this research are also discussed.

In Chapter 2, we introduce some theoretical background knowledge for this thesis. The message-passing algorithm and LDPC codes are explained. We also review some literature on LDPC codes over 1-D ISI channels and also the detection and equalization for 2-D ISI channels.

In Chapter 3, we investigate the problem of MLPD for 2-D ISI channels. The performance bounds of MLPD are derived and a low-complexity bounding technique is proposed. The effect of interference levels on the system performance is also examined.

In Chapter 4, we propose a JIDDS for discrete 2-D ISI channels with LDPC codes. The computational complexity of the scheme is analyzed and a reduced-state algorithm is proposed to reduce the complexity of the channel detector. Simulation results show that the proposed scheme mitigates the 2-D ISI effectively and outperforms some existing algorithms.

In Chapter 5, a JMPDA is proposed to implement the detection of 2-D ISI channel and decoding of LDPC code in a completely parallel manner. The LDPC code greatly improves the system performance while being well incorporated into the system, leading to a very high-speed decoder. Also in this chapter, we use the EXIT chart technique to analyze the convergence behavior of the proposed algorithm as well as the effects of

parameters on the system performance.

In Chapter 6, we first prove that the message-flow neighborhood of JMPDA will be tree-like for a sufficiently long code length. Based on the tree-like message-flow neighborhood, some concentration results are given and a modified density evolution algorithm is employed to calculate the threshold values of the JMPDA for LDPC coded 2-D ISI system.

In Chapter 7, we conclude our research works and provide some research topics for future research.

Chapter 2

Background Knowledge and Literature Review

In this chapter, we first present some background knowledge on the detection and decoding systems, such as the message-passing algorithm and low-density parity-check (LDPC) codes. Literature on LDPC codes over one-dimensional (1-D) inter-symbol interference (ISI) channels as well as detection and equalization for two-dimensional (2-D) ISI channels is reviewed.

2.1 Message-Passing Algorithm

Let $x_1, x_2, \dots, x_n$ be a collection of variables and each x_i takes on values in some domain A_i . Define $g(x_1, x_2, \dots, x_n)$ as a function of these variables that takes on certain values. Then there are n marginal functions $g_i(x_i)$, for $i = 1, 2, \dots, n$, associated with the function $g(x_1, \dots, x_n)$. For each $a \in A_i$, the value of $g_i(a)$ is obtained by summing $g(x_1, \dots, x_n)$ over all configurations of the variables $x_1, x_2, \dots, x_n$ such that $x_i = a$. Our target is to develop an efficient algorithm for computing marginal functions. Note that

Figure 2.1. Factor graph for the product $f_A(x_1)f_B(x_2)f_C(x_1, x_2, x_3)f_D(x_3, x_4)f_E(x_3, x_5)$.

$g(x_1, \dots, x_n)$ can be factored into a product of several local functions, each having some subset of $\{x_1, x_2, \dots, x_n\}$ as arguments, i.e.,

$$g(x_1, \dots, x_n) = \prod_{j \in J} f_j(X_j) \quad (2.1)$$

where J is a discrete index set, X_j is a subset of $\{x_1, x_2, \dots, x_n\}$, and $f_j(X_j)$ is a function with elements of X_j as arguments.

Example 2.1 : Let x_1, x_2, x_3, x_4 and x_5 be five variables. The joint function $g(x_1, x_2, x_3, x_4, x_5)$ of these five variables can be expressed as

$$g(x_1, x_2, x_3, x_4, x_5) = f_A(x_1)f_B(x_2)f_C(x_1, x_2, x_3)f_D(x_3, x_4)f_E(x_3, x_5) \quad (2.2)$$

where $J = \{A, B, C, D, E\}$, $X_A = \{x_1\}$, $X_B = \{x_2\}$, $X_C = \{x_1, x_2, x_3\}$, $X_D = \{x_3, x_4\}$, and $X_E = \{x_3, x_5\}$. Figure 2.1 depicts the factor graph corresponding to (2.2) in which the circles represent the variable nodes for each variable and the squares represent the factor nodes for each factor function.

2.1.1 Computing Marginal Functions

In many situations, we are interested in computing the marginal function $g_i(x_i)$ for some values of i . A straightforward way of obtaining the marginal function would be computing

$$g_i(x_i) = \sum_{\sim\{x_i\}} g(x_1, \dots, x_n) \quad (2.3)$$

for each i separately, where $\sim\{x_i\}$ denotes all variables $\{x_1, \dots, x_n\}$ except the variable x_i . The computation can be achieved by applying the distributive law. For example, the computation of $g_1(x_1)$ from Example 2-1 can be achieved via

$$\begin{aligned} g_1(x_1) &= \sum_{\sim\{x_1\}} g(x_1, x_2, x_3, x_4, x_5) \\ &= f_A(x_1) \left(\sum_{x_2} f_B(x_2) \left(\sum_{x_3} f_C(x_1, x_2, x_3) \cdot \left(\sum_{x_4} f_D(x_3, x_4) \right) \cdot \left(\sum_{x_5} f_E(x_3, x_5) \right) \right) \right) \\ &= f_A(x_1) \sum_{\sim\{x_1\}} \left(f_B(x_2) f_C(x_1, x_2, x_3) \left(\sum_{\sim\{x_3\}} f_D(x_3, x_4) \right) \left(\sum_{\sim\{x_3\}} f_E(x_3, x_5) \right) \right). \end{aligned} \quad (2.4)$$

Similarly, $g_3(x_3)$ can be computed as

$$\begin{aligned} g_3(x_3) &= \left(\sum_{\sim\{x_3\}} f_A(x_1) f_B(x_2) f_C(x_1, x_2, x_3) \right) \\ &\quad \cdot \left(\sum_{\sim\{x_3\}} f_D(x_3, x_4) \right) \left(\sum_{\sim\{x_3\}} f_E(x_3, x_5) \right). \end{aligned} \quad (2.5)$$

However, if we compute all the marginal functions separately by applying the above method, the approach is not efficient, since many of the subcomputations for different values of i will be the same. For example, there is a common term $\left(\sum_{\sim\{x_3\}} f_D(x_3, x_4) \right) \cdot$

$\left(\sum_{\sim\{x_3\}} f_E(x_3, x_5)\right)$ in the computations of both $g_1(x_1)$ and $g_3(x_3)$. Thus, it will be more efficient to compute all marginal functions by overlaying on a factor graph and sharing the intermediate values of subcomputations.

2.1.2 Sum-Product Algorithm

A generalized message-passing algorithm, called the sum-product algorithm [13], is an efficient method in computing the marginal functions. The sum-product algorithm operates according to the following rule.

The Sum-Product Update Rule: The message sent from a node v on an edge e is the product of the local function at v (or the unit function if v is a variable node) with all message received at v on edges other than e , summarized for the variable associated with e .

Let $\mu_{x \rightarrow f}(x)$ denote the message passed from a variable node x to a function node f and let $\mu_{f \rightarrow x}(x)$ denote the message sent from a function node f to a variable node x . Let $n(v)$ denote the set of neighbors of a given node v in a factor graph. Then, according to the sum-product algorithm, the message computations can be expressed as follows:

Variable to local function:

$$\mu_{x \rightarrow f}(x) = \prod_{g \in n(x) \setminus \{f\}} \mu_{g \rightarrow x}(x). \quad (2.6)$$

Local function to variable:

$$\mu_{f \rightarrow x}(x) = \sum_{\sim\{x\}} \left(f(X) \prod_{y \in n(f) \setminus \{x\}} \mu_{y \rightarrow f}(y) \right) \quad (2.7)$$

where $X = n(f)$ is the set of arguments of the function f . The update rule can be

summarized as the message sent from node u to node v should be extrinsic in the sense that it should include all available information to u other than the information from v .

In the following, we will compute the marginal functions based on Example 2-1 to illustrate the detailed sum-product algorithm.

Step 1:

$$\begin{aligned}\mu_{f_A \rightarrow x_1}(x_1) &= \sum_{\sim\{x_1\}} f_A(x_1) = f_A(x_1) \\ \mu_{f_B \rightarrow x_2}(x_2) &= \sum_{\sim\{x_2\}} f_B(x_2) = f_B(x_2) \\ \mu_{x_4 \rightarrow f_D}(x_4) &= 1 \\ \mu_{x_5 \rightarrow f_E}(x_5) &= 1.\end{aligned}$$

Step 2:

$$\begin{aligned}\mu_{x_1 \rightarrow f_C}(x_1) &= \mu_{f_A \rightarrow x_1}(x_1) \\ \mu_{x_2 \rightarrow f_C}(x_2) &= \mu_{f_B \rightarrow x_2}(x_2) \\ \mu_{f_D \rightarrow x_3}(x_3) &= \sum_{\sim\{x_3\}} \mu_{x_4 \rightarrow f_D}(x_4) f_D(x_3, x_4) \\ \mu_{f_E \rightarrow x_3}(x_3) &= \sum_{\sim\{x_3\}} \mu_{x_5 \rightarrow f_E}(x_5) f_E(x_3, x_5).\end{aligned}$$

Step 3:

$$\begin{aligned}\mu_{f_C \rightarrow x_3}(x_3) &= \sum_{\sim\{x_3\}} \mu_{x_1 \rightarrow f_C}(x_1) \mu_{x_2 \rightarrow f_C}(x_2) f_C(x_1, x_2, x_3) \\ \mu_{x_3 \rightarrow f_C}(x_1) &= \mu_{f_D \rightarrow x_3}(x_3) \mu_{f_E \rightarrow x_3}(x_3).\end{aligned}$$

Step 4:

$$\begin{aligned}
\mu_{f_C \rightarrow x_1}(x_1) &= \sum_{\sim\{x_1\}} \mu_{x_3 \rightarrow f_C}(x_3) \mu_{x_2 \rightarrow f_C}(x_2) f_C(x_1, x_2, x_3) \\
\mu_{f_C \rightarrow x_2}(x_2) &= \sum_{\sim\{x_1\}} \mu_{x_3 \rightarrow f_C}(x_3) \mu_{x_1 \rightarrow f_C}(x_1) f_C(x_1, x_2, x_3) \\
\mu_{x_3 \rightarrow f_D}(x_3) &= \mu_{f_C \rightarrow x_3}(x_3) \mu_{f_E \rightarrow x_3}(x_3) \\
\mu_{x_3 \rightarrow f_E}(x_3) &= \mu_{f_C \rightarrow x_3}(x_3) \mu_{f_D \rightarrow x_3}(x_3).
\end{aligned}$$

Step 5:

$$\begin{aligned}
\mu_{x_1 \rightarrow f_A}(x_1) &= \mu_{f_C \rightarrow x_1}(x_1) \\
\mu_{x_2 \rightarrow f_B}(x_2) &= \mu_{f_C \rightarrow x_2}(x_2) \\
\mu_{f_D \rightarrow x_4}(x_4) &= \sum_{\sim\{x_4\}} \mu_{x_3 \rightarrow f_D}(x_3) f_D(x_3, x_4) \\
\mu_{f_E \rightarrow x_5}(x_5) &= \sum_{\sim\{x_5\}} \mu_{x_3 \rightarrow f_E}(x_3) f_E(x_3, x_5).
\end{aligned}$$

Termination:

$$\begin{aligned}
g_1(x_1) &= \mu_{f_A \rightarrow x_1}(x_1) \mu_{f_C \rightarrow x_1}(x_1) \\
g_2(x_2) &= \mu_{f_B \rightarrow x_2}(x_2) \mu_{f_C \rightarrow x_2}(x_2) \\
g_3(x_3) &= \mu_{f_C \rightarrow x_3}(x_3) \mu_{f_D \rightarrow x_3}(x_3) \mu_{f_E \rightarrow x_3}(x_3) \\
g_4(x_4) &= \mu_{f_D \rightarrow x_4}(x_4) \\
g_5(x_5) &= \mu_{f_E \rightarrow x_5}(x_5).
\end{aligned}$$

2.2 Low-Density Parity-Check Codes

During the past few decades, a lot of research has been done in searching for good error-correcting codes (ECCs). Turbo codes, proposed by Berrou et al. in [14] and [15], exhibit superior capabilities in error correcting and achieve a BER of 10^{-5} at a value of E_b/N_0 0.7 dB, which is 0.7 dB away from the Shannon capacity for the unconstrained additive white Gaussian noise (AWGN) channel and only 0.5 dB away from the binary-input AWGN channel. Brink later constructed turbo codes with performance much closer to the binary Shannon limit [16]. Another class of ECCs, called low-density parity-check (LDPC) codes, was first proposed by Gallager in [17]. Inspired by the success of turbo codes, the potentials of LDPC codes were re-examined in the mid-1990's with the work of MacKay, Luby, and others [18]-[20]. Compared to turbo codes, the LDPC codes have lower decoding complexity and shorter latency. However, these regular LDPC codes perform about 0.5 dB worse than turbo codes. The construction of irregular LDPC codes was proposed by Luby et al. in [21], [22], which can be used to further improve the performance of LDPC codes. In [23], Richardson and Urbanke proposed a method, called density evolution (DE), to analyze the asymptotic performance of LDPC codes over various memoryless channels. By using this method to optimize the degree distributions of LDPC codes, it was shown in [24] that the LDPC codes can approach the Shannon limit by 0.0045 dB. In [25], a Gaussian approximation method was proposed which can simplify the analysis of the LDPC decoding algorithm.

2.2.1 LDPC Encoding

LDPC codes belong to a class of linear block codes which can be uniquely described by a parity-check matrix $\mathbf{H}_c$. Here for the sake of simplicity, we will only consider binary LDPC codes. The parity-check matrix $\mathbf{H}_c$ of an LDPC code has a low density of

1's which ensures good error-correcting performance and guarantees that the decoding process can be performed efficiently. The key of designing an LDPC code is the construction of an LDPC matrix with prescribed properties. Many construction methods can be found in the literature [19],[26]-[29].

Given an LDPC matrix $\mathbf{H}_c$, we need to compute the generator matrix $\mathbf{G}_c$. We first perform Gauss-Jordan elimination on $\mathbf{H}_c$ to obtain a new matrix $\mathbf{H}'_c$ in the form of

$$\mathbf{H}'_c = [\mathbf{P}|\mathbf{I}_{N-K}] \quad (2.8)$$

where N is the length of the LDPC code, K is the length of the input bit message, $\mathbf{P}$ is a matrix of size $(N - K) \times K$, and $\mathbf{I}_{N-K}$ is a $N - K$ identity matrix. The generator matrix $\mathbf{G}_c$ can therefore be obtained via

$$\mathbf{G}_c = [\mathbf{I}_K|\mathbf{P}^T] \quad (2.9)$$

where the superscript T denotes the transpose operation. The parity-check matrix $\mathbf{H}_c$ and the generator matrix $\mathbf{G}_c$ have a property that $\mathbf{H}_c \cdot \mathbf{G}_c^T = \mathbf{0}$. The code word $\mathbf{s}$ is generated by multiplying the message $\mathbf{u}$ by $\mathbf{G}_c$ in binary field as

$$\mathbf{s} = \mathbf{u}\mathbf{G}_c. \quad (2.10)$$

2.2.2 LDPC Decoding

The decoding of LDPC code is a process in computing the *a posteriori* probability (APP) of the transmitted bits given the received information of the LDPC decoder. In most circumstances, the received information is soft information *per se* or can be converted to soft information. For example, we consider an AWGN channel, let $c_i =$

$2s_i - 1$ be the i -th transmitted binary value. Then the i -th received value is $y_i = c_i + n_i$ where n_i is the independent and Gaussian distributed noise with zero mean and variance of σ^2 . It can be shown that the soft information associated with the i -th received value is [30]

$$\Pr(c_i = c|y_i) = [1 + \exp(-2cy_i/\sigma^2)]^{-1} \quad (2.11)$$

where $c \in \{\pm 1\}$. We can also use factor graph to describe the LDPC codes. There are two types of nodes in the factor graph representation of LDPC codes, namely, variable nodes and check nodes. This graph is also called Tanner graph [31]. Each variable node corresponds to a bit variable, thus there are N variable nodes. Each row of the parity-check matrix $\mathbf{H}_c$ corresponds to a parity-check equation, thus there are $N - K$ check nodes. The i -th check node is connected to the j -th variable node if the i -th row and j -th column element $h_{i,j}$ of the parity-check matrix equals to 1.

Example 2.2 : Consider a $(10, 5)$ LDPC code with column weight 2 and row weight 4 whose parity-check matrix is

$$\mathbf{H}_c = \begin{bmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}. \quad (2.12)$$

The corresponding Tanner graph of this LDPC code is shown in Fig. 2.2. In this figure, the circles represent the variable nodes associated with each bit s_i . The squares represent the check nodes associated with the parity-check constraints and they are denoted as f_i . Different from the factor graph in Fig. 2.1, the factor graph in Fig. 2.2

Figure 2.2. Tanner graph for a typical LDPC code.

contains cycles which is shown as bold edges in the figure. In this example, the LDPC code has cycles of length 6. Due to the existence of cycles, the information transferred between the variable nodes and check nodes is not perfectly extrinsic. Short cycles are very harmful to the performance of LDPC codes. Although the factor graph of LDPC codes has cycles, the sum-product update rule still works well in instructing the information transfer between the variable nodes and check nodes. The decoding algorithm is summarized as follows:

Initialization: For $i = 1, 2, \dots, N$, set $P_i = \Pr(c_i = 1|y_i)$. Then for all i, j for which $h_{ij} = 1$ set $z_{ij}(1) = P_i$ and $z_{ij}(0) = 1 - P_i$, where $z_{i,j}(s)$ is the probability that $s_i = s$ which is sent from the i -th variable node to the j -th check node.

Check node update: Let $q_{ji}(s)$ be the probability that check equation is satisfied given $s_i = s$ which is sent from the j -th check node to the i -th variable node. Then $q_{ji}(s)$ can be updated via

$$q_{ji}(0) = \frac{1}{2} + \frac{1}{2} \prod_{i' \in D_c(j) \setminus i} (1 - 2z_{i'j}(1)) \quad (2.13)$$

and

$$q_{ji}(1) = 1 - q_{ji}(0) \quad (2.14)$$

where $D_c(j) \setminus i$ represents the set of variable nodes connected to the j -th check node except for the i -th variable node.

Variable node update: The information sent from variable nodes to check nodes can be updated via

$$z_{ij}(0) = K_{ij}(1 - P_i) \prod_{j' \in D_v(i) \setminus j} q_{j'i}(0) \quad (2.15)$$

and

$$z_{ij}(1) = K_{ij}P_i \prod_{j' \in D_v(i) \setminus j} q_{j'i}(1) \quad (2.16)$$

where K_{ij} is a scale parameter to ensure that $z_{ij}(0) + z_{ij}(1) = 1$, $D_v(i) \setminus j$ represents the set of check nodes connected to the i -th variable node except for the j -th check node. The above check node update and variable node update are repeated for several times. Finally, the APP of the transmitted bit is given by

$$z_i(0) = K_i(1 - P_i) \prod_{j \in D_v(i)} q_{ji}(0) \quad (2.17)$$

and

$$z_i(1) = K_iP_i \prod_{j \in D_v(i)} q_{ji}(1) \quad (2.18)$$

where $z_i(0)$ and $z_i(1)$ represent the APPs that $s_i = 0$ and $s_i = 1$, respectively, $D_v(i)$ represents the set of check nodes connected to the i -th variable node, and K_i is chosen to ensure that $z_i(0) + z_i(1) = 1$.

2.3 LDPC Codes Over 1-D ISI Channels

For one-dimensional (1-D) inter-symbol interference (ISI) channel, if there is no noise, ISI can be perfectly eliminated by passing the readback signal through an inverse filter whose frequency response is the inverse of the channel response. However, in practice, noise always exists. When passed through the inverse filter, the noise in the readback signal can be easily amplified. Therefore, it is more desirable to detect the 1-D ISI using other methods, such as the Viterbi algorithm (VA) [32] and Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [33]. Both the VA and BCJR algorithms work on trellis structure whose complexity is proportional to the length of ISI. At high recording densities, the ISI length can be so large that the VA and BCJR algorithms become too complicated to be implemented. In such scenarios, a partial-response (PR) equalizer is required to first reduce the length of ISI extent. A PR channel can be represented by a PR polynomial

$$h(D) = \sum_{i=0}^L h_i D^i \quad (2.19)$$

where D denotes one-bit delay, L is the ISI length of the PR channel. Let c_i denote the input bit sequence to the PR channel and n_i denote AWGN of the channel. The output of PR channel can be expressed as

$$y_m = \sum_{i=0}^L h_i c_{m-i} + n_m. \quad (2.20)$$

Then the VA or BCJR algorithm is applied to the output of the PR equalizer to remove the existing ISI. The performance bounds of 1-D ISI channels were studied in [34]-[37]. Various bounds on the capacity of 1-D ISI channel were developed for certain 1-D ISI channels in [38]-[40].

Following the introduction of turbo codes, an iterative detection and decoding

Figure 2.3. Block diagram of LDPC coded ISI channel with turbo equalization.

scheme was developed and has achieved much success [41],[42]. In iterative detection and decoding schemes, the ISI channel is considered as the inner code and the error correcting codes (ECCs) serve as the outer code. The soft-input soft-output (SISO) detector and decoder exchange extrinsic information and work iteratively in the same way as inner and outer decoders of turbo codes. Thus, this scheme is called the turbo equalization [43],[44]. Since VA only produces hard decisions, BCJR algorithm and soft-output Viterbi algorithm (SOVA) [45] are used for the SISO detection. LDPC code is ideal for the SISO decoder due to its capability of error correcting and soft decoding. The application of LDPC codes to 1-D ISI channels has been extensively studied in [46]-[56].

Figure 2.3 depicts the block diagram of a turbo equalization system, which consists of the LDPC encoder, the PR equalized channel, the channel detector, and the LDPC

decoder. The information exchanged between the SISO detector and decoder must be extrinsic information which corresponds to newly generated information. The extrinsic information is often expressed in the form of log likelihood ratio (LLR) which is defined as the logarithm of the ratio of the probability that a sample corresponds to 1 to the probability that the sample corresponds to -1. Denote $L_c(c_i)$ and $L_l(c_i)$ as the LLRs corresponding to c_i of the output of channel detector and LDPC decoder, respectively. Then the extrinsic LLR transferred from the channel detector to the LDPC decoder is equal to *a posteriori* LLR minus *a priori* LLR, which is computed as

$$L_{ext}^1(c_i) = L_c(c_i) - L_{ext}^2(c_i) \quad (2.21)$$

where $L_{ext}^2(c_i)$ is the extrinsic LLR transferred from the LDPC decoder to the channel detector. Similarly, $L_{ext}^2(c_i)$ is computed as

$$L_{ext}^2(c_i) = L_l(c_i) - L_{ext}^1(c_i). \quad (2.22)$$

The above extrinsic information is used to update the *a priori* information in the channel detector and LDPC decoder. To achieve better performance, the exchange of extrinsic information between the channel detector and the LDPC decoder can be repeated for several times. Finally, the LDPC decoder produces the estimated bit $\hat{u}_i$.

Some joint detection and decoding or simply joint decoding schemes were proposed in [52]-[56] to implement channel detection and LDPC decoding simultaneously. In the channel detector, the BCJR algorithm or SOVA serially operates on the channel trellis. In the LDPC decoder, the sum-product algorithm operates on the Tanner graph in parallel. The idea of these joint decoding schemes is to design some message-passing decoding algorithms that operate on consolidated graphs, which combine the

channel constraints and parity-check constraints. Therefore, the advantage of parallel implementation of the LDPC decoder can be used to reduce the overall decoding delay.

Some works on the theoretical analysis of turbo equalization for 1-D ISI channels can be found in [57]-[64]. In [57], the SISO linear equalizer and its application to the design of turbo equalization systems were studied by tracking mutual information evolution in extrinsic information transfer (EXIT) chart instead of extensive simulation. In [58] and [59], the authors analyzed the convergence behavior of the turbo equalization using EXIT chart. When monitoring the status of the iterative process, the number of iterations can be effectively adjusted, which can lower the overall complexity. The performance bounds for high rate linear codes over PR channels were derived in [60]. Following this, performance bounds for LDPC codes over PR channels were given in [61]. In [62], the DE technique was adopted to analyze the message error probability in LDPC coded ISI channels. The authors proved that there exist thresholds for LDPC coded ISI channels and the threshold values were calculated using a modified DE algorithm. Based on [62], the authors modified the result to account for symbol errors in [63]. In [64], the thresholds and scheduling of LDPC codes over ISI channels were investigated using the method of Gaussian approximation.

2.4 Detection and Equalization for 2-D ISI Channels

In ultra-high density magnetic recording systems, the read head senses not only the desired signal with ISI in the down-track direction but also inter-track interference (ITI) in the cross-track direction. Therefore, ISI together with ITI give rise to a two-dimensional (2-D) interference channel model. The modeling of the 2-D pulse response

and the readback signal has been studied in [65]-[71]. Here, we only consider the equivalent discrete time 2-D interference channel which can be represented by a channel response matrix

$$\mathbf{H} = \begin{bmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,N_h-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,N_h-1} \\ \dots & \dots & & \dots \\ h_{M_h-1,0} & h_{M_h-1,1} & \dots & h_{M_h-1,N_h-1} \end{bmatrix} \quad (2.23)$$

where M_h and N_h are the numbers of magnetic islands the read head senses in the cross-track and down-track directions, respectively. The readback signal $r_{i,j}$ can be expressed as

$$r_{i,j} = \sum_{m=0}^{M_h-1} \sum_{n=0}^{N_h-1} h_{m,n} \cdot x_{i-m,j-n} + v_{i,j} \quad (2.24)$$

where $x_{i,j}$ is the recorded information bit and we assume the number of rows and columns of the 2-D array $\mathbf{X} = \{x_{i,j} : i, j\}$ are N_r and N_c , respectively. The variable $v_{i,j}$ represents AWGN. The capacity of the channel is the supremum of the mutual information rates over all input distribution $\mathbf{X}$

$$C = \sup_{\mathbf{X}} I(\mathbf{X}; \mathbf{R}). \quad (2.25)$$

When the channel inputs are independent and identically distributed (i.i.d.) equiprobable binary bits, the above mutual information rate is called the symmetric information rate (SIR) which can be interpreted as the theoretical limits on the achievable storage densities.

In [72], a simulation-based method was introduced to estimate the information rate

of finite-state 1-D ISI channels. The idea underlying this method is to estimate the channel output entropy by computing the joint probability of a long channel output realization. The calculation is facilitated by following the forward sum-product recursion on the joint trellis that combines the source and channel constraints. However, this approach cannot be easily extended to 2-D ISI channels, since there is no simple 2-D detection algorithm like the forward recursion of BCJR algorithm to compute the probability of a large channel output array realization. In [73], the authors overcame this problem by using conditional entropies of smaller output arrays as the basis for lower and upper bounds on the entropy rate of a large output array. By modifying the 1-D Monte Carlo technique to the calculation of these conditional entropies, tight upper and lower bounds on SIR were computed for 2-D ISI channels with small impulse response support. In [74], the authors extended the 1-D bounds in [38] and [39] and proposed a more general and tractable approach to deriving bounds on the SIR of 2-D ISI channels with larger ISI span. In [75], the authors proposed another approach to estimate the SIR of 2-D ISI channels. This approach is established upon a connection between the Shannon-theoretic information rate and the free energy. The free energy is tracked by approximating it using the cluster variation method, implemented via generalized belief propagation. This method is shown to provide a practically accurate estimate of the information rate of 2-D ISI channels.

2.4.1 Optimal Detection

It is very simple to think of a detection method that tests all the combinations of data bits and chooses one which has the largest probability. Although this could produce the best performance, the complexity is $2^{N_r N_c}$ which is too complicated to be implemented. Optimal detection, such as maximum likelihood (ML) or maximum *a posteriori* prob-

Figure 2.4. Markov process for 1-D ISI channel.

ability (MAP) for 2-D ISI channels is computationally prohibitive due to the lack of simple graph description of 2-D ISI channels.

For a 1-D ISI channel, it is easy to group the adjacent strings of L bits to form a 1st order Markov process over the larger alphabet as shown in Fig. 2.4. The region **B** separates the sequence of bits into two independent regions **A** and **C**. The length of **B** is only dependent on the interference length L and it is not related to the length of the input data sequence. For a 2-D ISI channel, we must condition on a bounding strip to create 2 conditionally independent regions. The bounding region must completely isolate the 2 regions to avoid unaccounted-for interaction via some paths [76]. Figure 2.5 depicts an example of bounding regions in which the regions **A** and **C** are conditionally independent, such that $P(\mathbf{A}, \mathbf{C} | \mathbf{B}) = P(\mathbf{A} | \mathbf{B})P(\mathbf{C} | \mathbf{B})$. It can be observed that to decode the entire 2-D data array, the bounding region must at some point span the entire height or width of the array. For example, one approach to optimal detection is to treat each column as a symbol consisting of N_r bits and use 1-D optimal detection algorithm (Viterbi/BCJR) to detect each symbol over a larger alphabet in the row-wise direction as shown in Fig. 2.6. But the computational complexity of this detector is still in the order of $2^{N_r M_h}$. Thus, the complexity of the optimal detector grows exponentially with the size of the 2-D data array. When the size of the data array is large, the complexity becomes prohibitively high. It has been proved that the ML detection for a 2-D ISI channel is NP-complete [77].

Figure 2.5. 2-D Markov random field with bounding strip **B** making regions **A** and **C** conditionally independent.

Figure 2.6. Optimal 2-D detection by using 1-D detection algorithm over a larger alphabet.

2.4.2 Suboptimal Equalization and Detection

To reduce the complexity of optimal 2-D detection, various suboptimal equalization and detection algorithms were proposed in [78]-[93]. In [78], the authors assumed symmetric ITI and proposed a modified VA to mitigate the effect of ITI. The readback signal is first passed through a 1-D equalizer and then detected by the modified Viterbi detector.

The modified VA takes into account the contribution of ITI and works on a complicated trellis with more branches entering and leaving each state. In [79] and [80], the authors proposed and investigated a 2-D generalized partial response (2-D GPR) equalizer. In their work, the target of the 2-D GPR is chosen to force the effect of ITI to zero. Thus, after the equalization, the signal can be approximated as the output of a 1-D ISI channel and a Viterbi detector is followed to further detect the signal. To reduce the complexity of this 2-D GPR equalizer, Keskinöz designed another 2-D GPR equalizer that reduces the effect of both ITI in the cross-track direction and ISI in the down-track direction [81]. The equalizer proposed by Keskinöz does not incorporate any zero forcing constraint and this simplifies the calculations in the equalizer. In [82], the author proposed a modified version of this equalizer which generates soft feedback during the iterations inside the equalizer. In [83], a suboptimal strip-wise algorithm (SSWA) was proposed with the name of MAP algorithm. In SSWA, a modified BCJR algorithm first computes the probability of each symbol which is a column-wise 3 bits vector. Then based on these probabilities, the bit sequence in the middle track is detected. This SSWA is not a MAP algorithm for 2-D detection, since it does not process the whole received data page to detect the recorded data. It is an optimal detection algorithm only when signals of three adjacent tracks are available for detection. A joint-track equalization technique was also proposed in [83] based on the multi-input single-output system model [66]. Unlike the equalizers described in [9] and [78] in which only the main track was equalized to a 1-D target, this equalizer equalized all three tracks to a 2-D target. It has been shown that this joint-track equalizer performs close to the SSWA.

Some iterative detection algorithms were proposed in [84]-[87]. In [84], an iterative detection and decoding algorithm was proposed for separable 2-D ISI channels. In this work, the separable 2-D interference channel is modeled as a concatenation of row ISI

and column ISI that are further dealt with in row detector and column detector, respectively. However, this detection algorithm works only when the 2-D channel response matrix is separable, but this requirement is not always satisfied in practice. A similar work was proposed in [85]. In this work, the authors proposed an iterative 2-D SOVA which consists of two 1-D SOVAs working iteratively in two directions to remove the ISI and ITI. In [86] and [87], an iterative row-column soft-decision feedback algorithm (IRCSDFFA) was proposed for the detection of signals over a 2-D interference channel. This detection algorithm is a concatenation of two SISO detectors. The soft information exchanged between the two detectors is scaled down by a weight factor which is normally a function of the iteration number.

The application of LDPC codes has also been studied in [88]-[93]. In [88], a minimum mean-squared-error (MMSE)-LDPC decoding scheme was proposed. The equalizer used in this scheme was a MMSE equalizer [89] which calculates the linear MMSE estimate of each recorded bit given a finite-size array of the readback data. The equalizer generates probabilities of the recorded bits and exchanges soft information iteratively with the LDPC decoder. In [90], a message-passing based scheme for joint equalization and decoding for a 2-D ISI channel was proposed. This scheme performs sum-product message-passing decoding on a joint factor graph of the LDPC code and the channel. In [91], the application of non-binary LDPC codes for 2-D ISI channels was investigated based on the “multiple islands per read head” model [66]. In this model, the read head is assumed to span several sub-tracks and the bits from all the sub-tracks that the read head senses are treated as a symbol. Thus a 1-D maximum-likelihood sequence detection (MLSD) is applied and works in the down-track direction to detect the symbols. However, due to the indistinguishable symbol problem, this model results in serious code rate loss (up to 1/2) and the application of non-binary LDPC will further reduce the code rate. This is especially intolerable in magnetic recording

Figure 2.7. Iterative detection and decoding for separable 2-D ISI channels.

systems where the code rate is a major concern. The iterative detection and decoding has also been studied in [84], [92] and [93]. In [84], the LDPC decoder exchange LLR information with one of the two constituent detectors as shown in Fig. 2.7. A Markov chain Monte Carlo (MCMC) based 2-D detection algorithm was proposed in [92]. In this work, the *a posteriori* probability of the desired binary bit is converted to the expected value of a function. Using the Monte Carlo techniques, the calculation of the expectation is approximated by taking an average of the samples. As the MCMC detection algorithm works on binary bits, LDPC codes are conveniently incorporated to the detection system as that in a 1-D ISI channel as shown in Fig. 2.3. In [93], the LDPC coded codeword is split into three tracks and the LDPC code constraint provides partial estimations of ITI based on which a modified SOVA detects the 2-D interfered bit.

As briefly described in Chapter 1, signals in magnetic channel are read by detecting the transitions in magnetizations, which depend on the signal amplitude more fundamentally than the signal power. However, in 2-D interference channel, the magnetic signal of a certain bit not only exists in its own readback signal but also contributes to the readback signals of the adjacent bits. In order to take into account the amount of all the available signals that can be potentially made use of, we define the signal-to-noise

ratio (SNR) as

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum_{i=0}^{M_h-1} \sum_{j=0}^{N_h-1} (h_{i,j})^2}{2R\sigma^2} \right) \text{ dB} \quad (2.26)$$

where R is the overall code rate and σ^2 is the variance of the AWGN. The above SNR definition can be comprehended as the energy per user bit over the one-sided power spectral density of the AWGN. This definition of SNR is widely used in various technical articles [84],[86],[87],[92],[93] in which the authors tried to fully make use of the signal power in two dimensions. We also adopt this definition of SNR throughout this thesis.

Chapter 3

Maximum-Likelihood Page Detection

In this chapter, we study the problem of maximum-likelihood page detection (MLPD) for two-dimensional (2-D) interference channels. For one-dimensional (1-D) intersymbol interference (ISI) channels [35], the Viterbi algorithm is the commonly used maximum-likelihood detection algorithm. However, as argued in Chapter 2, there is no simple maximum-likelihood detection algorithm for 2-D interference channels. Nevertheless, the investigation of MLPD for 2-D interference channels is valuable, as the characterization of MLPD can provide an essential understanding of 2-D interference channels. Moreover, the optimal detection performance of MLPD can be used to assess the performance of suboptimal detectors for mitigating 2-D interference. The performance bounds of MLPD have been given in [94]. In this chapter, we derive the performance bounds of MLPD in a different way. The purpose of doing this is: The new way of deriving the bounds can provide a new understanding of the 2-D interference channels. We also go further to propose an approximated bounding method which can be used to accurately and quickly compute the performance bounds. In addition, based on our

derived performance bounds, a parametric effect of the 2-D interference level is studied.

3.1 Channel Model and MLPD

It has been shown that the ITI is mainly caused by one adjacent track on each side of the main track and the ITI from more than one outer track can be ignored without causing much performance loss [88]. Under this assumption, a channel response matrix of size $3 \times N_h$ can be constructed. However, in this chapter, we consider a more general channel whose channel response matrix has a size of $M_h \times N_h$ as

$$\mathbf{H} = \begin{bmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,N_h-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,N_h-1} \\ \dots & \dots & \dots & \dots \\ h_{M_h-1,0} & h_{M_h-1,1} & \dots & h_{M_h-1,N_h-1} \end{bmatrix} \quad (3.1)$$

The desire signal is interfered by $N_h - 1$ symbols in the down-track direction and $M_h - 1$ tracks in the cross-track direction, which can be expressed as

$$\begin{aligned} y_{i,j} &= h_{i,j} * x_{i,j} \\ &= \sum_{m=0}^{M_h-1} \sum_{n=0}^{N_h-1} h_{m,n} \cdot x_{i-m,j-n} \end{aligned} \quad (3.2)$$

where $x_{i,j}$ is the recorded data chosen independently and uniformly from a finite alphabet A with an alphabet size of M_a . The input data $\{x_{i,j} \in A, (i,j) \in \mathcal{X}\}$ is assumed to span the region $\mathcal{X}$ and is equal to zero if $i \leq 0$ or $j \leq 0$. Hence, the received signal consists of the 2-D interfered signal plus AWGN, which can be expressed as

$$r_{i,j} = y_{i,j} + v_{i,j} \quad (3.3)$$

where $v_{i,j}$ represents the AWGN component with zero mean and variance of σ^2 . We use a boldface character to denote the entire 2-D page of the corresponding signals, e.g., $\mathbf{R}$ represents the whole received signals with $\mathbf{R} = \{r_{i,j}, (i,j) \in \mathcal{X}\}$.

The MLPD finds the whole input data $\tilde{\mathbf{X}} = \{\tilde{x}_{i,j}, (i,j) \in \mathcal{X}\}$ such that the probability density $p[\mathbf{R}|\tilde{\mathbf{X}}]$ is maximized. Let $\tilde{\mathbf{Y}} = \{\tilde{y}_{i,j}, (i,j) \in \mathcal{X}\}$ represent the 2-D interfered data corresponding to $\tilde{\mathbf{X}}$. Due to the one to one mapping between $\tilde{\mathbf{X}}$ and $\tilde{\mathbf{Y}}$, maximizing $p[\mathbf{R}|\tilde{\mathbf{X}}]$ is equivalent to maximizing $p[\mathbf{R}|\tilde{\mathbf{Y}}]$. Since the noise terms $v_{i,j}$ are assumed to be independent of each other, the log likelihood function $\ln p[\mathbf{R}|\tilde{\mathbf{Y}}]$ breaks up into a sum of terms and can be expressed as

$$\ln p[\mathbf{R}|\tilde{\mathbf{Y}}] = \sum_{(i,j) \in \mathcal{X}} \ln p_n[r_{i,j} - \tilde{y}_{i,j}] \quad (3.4)$$

where $p_n(\cdot)$ represents the probability density function (pdf) of the noise component $v_{i,j}$. By substituting $p_n(\cdot)$ with the Gaussian pdf, it can be shown that MLPD is equivalent to finding the data page that minimizes the function $\Gamma_{\mathcal{X}}(\tilde{\mathbf{X}})$. Note that $\Gamma_{\mathcal{X}}(\tilde{\mathbf{X}})$ is dependent on the received signals $\mathbf{R}$ and the estimated array $\tilde{\mathbf{X}}$ over a certain region $\mathcal{X}$ and is defined as

$$\Gamma_{\mathcal{X}}(\tilde{\mathbf{X}}) = \sum_{(i,j) \in \mathcal{X}} [r_{i,j} - \tilde{y}_{i,j}]^2. \quad (3.5)$$

Furthermore, let the most likely data page be denoted by $\hat{\mathbf{X}}$ which is assumed to be obtained by an exhaustive search algorithm.

3.2 Probability of Error

3.2.1 Independent Error Event

An error event $\mathbf{E}$ is defined as the difference between an input data array $\mathbf{X}_1$ and the corresponding estimated data array $\hat{\mathbf{X}}_1$ over a certain region $\mathcal{E}$:

$$\mathbf{E} = \mathbf{X}_1 - \hat{\mathbf{X}}_1 = \{e_{i,j}, (i,j) \in \mathcal{E}\}. \quad (3.6)$$

The full-page error event is thus defined as $\mathbf{E}_\mathbf{X} = \mathbf{X} - \hat{\mathbf{X}}$. Two error events are independent of each other if there are large enough zero regions separating the two error events such that any decision made in one error event cannot affect the decision of the other event. Equivalent, in mathematical expressions, error events $\mathbf{E}_1 = \{e_{i_1,j_1}, (i_1,j_1) \in \mathcal{E}_1\}$ and $\mathbf{E}_2 = \{e_{i_2,j_2}, (i_2,j_2) \in \mathcal{E}_2\}$ are independent of each other if $\forall (i_1,j_1) \in \mathcal{E}_1, e_{i_1,j_1} \neq 0$ and $\forall (i_2,j_2) \in \mathcal{E}_2, e_{i_2,j_2} \neq 0$, it satisfies the condition $|i_1 - i_2| \geq M_h$ or $|j_1 - j_2| \geq N_h$. An error event is an independent error event if it is independent of all the other error events and it cannot be further divided into any error events that are independent of each other. The definition of the independent error event in a 2-D interference channel can be viewed as an extension of that in a 1-D ISI channel [35]. In Figure 3.1, we show an example of three independent error events. In this figure, the size of the channel response matrix is assumed to be 3×3 .

3.2.2 Probability of a Particular Independent Error Event

Based on the above definition, since any decision made in an independent error event cannot affect the decisions in other error events, we can now focus on a particular independent error event $\mathbf{E}$ that is assumed to start from the point (k_1, l_1) and spans the region $\mathcal{E}$.

Figure 3.1. An example of three independent error events.

For the occurrence of $\mathbf{E}$, the following two associated subevents must occur:

$\mathbf{E}'$: Over the region $\mathcal{E}$, the input array $\mathbf{X}_{\mathcal{E}}$ must be such that $\mathbf{X}_{\mathcal{E}} + \mathbf{E}$ is an allowable array. For example, if the inputs are modulated to be integers in the range $0 \leq x_{i,j} \leq M_a - 1$, when $e_{i,j} = q$ only $(M_a - q)$ possible values of $x_{i,j}$ are permissible. Assuming the input symbols are independent of each other, we have

$$\Pr(\mathbf{E}') = \prod_{(i,j) \in \mathcal{E}} \frac{M_a - |e_{i,j}|}{M_a}. \quad (3.7)$$

$\mathbf{E}''$: The noise terms $v_{i,j}$, $(i,j) \in \mathcal{E}$, must be such that the estimated array $\hat{\mathbf{X}}_{\mathcal{E}}$ given

by MLPD has greater likelihood than $\mathbf{X}_{\mathcal{E}}$ which can be expressed as

$$\begin{aligned}\Gamma_{\mathcal{E}}(\hat{\mathbf{X}}_{\mathcal{E}}) &\leq \Gamma_{\mathcal{E}}(\mathbf{X}_{\mathcal{E}}) \\ \sum_{(i,j) \in \mathcal{E}} [r_{i,j} - \hat{y}_{i,j}]^2 &\leq \sum_{(i,j) \in \mathcal{E}} [r_{i,j} - y_{i,j}]^2.\end{aligned}\quad (3.8)$$

By substituting $r_{i,j} = y_{i,j} + v_{i,j}$, $y_{i,j} = h_{i,j} * x_{i,j}$ and $\hat{y}_{i,j} = h_{i,j} * (x_{i,j} + e_{i,j})$, the above inequality can be rewritten as

$$\begin{aligned}\sum_{(i,j) \in \mathcal{E}} [v_{i,j} - h_{i,j} * e_{i,j}]^2 &\leq \sum_{(i,j) \in \mathcal{E}} [v_{i,j}]^2 \\ \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}]^2 &\leq 2 \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}] \cdot v_{i,j}.\end{aligned}\quad (3.9)$$

On the right hand side of the above inequality, $\{v_{i,j}, (i,j) \in \mathcal{E}\}$ are independent AWGN components. Thus, the term $2 \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}] \cdot v_{i,j}$, which represents the weighted sum of AWGN components, is still Gaussian distributed with mean of

$$\mathbb{E} \left(2 \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}] \cdot v_{i,j} \right) = 0 \quad (3.10)$$

and variance of

$$\mathbb{E} \left(\left(2 \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}] \cdot v_{i,j} \right)^2 \right) = 4\sigma^2 \cdot \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}]^2, \quad (3.11)$$

where the notation $\mathbb{E}(\cdot)$ represents the expectation operation.

Let the error event distance $d(\mathbf{E})$ be denoted as the square root of $d^2(\mathbf{E})$ which is

defined as

$$d^2(\mathbf{E}) = \sum_{(i,j) \in \mathcal{E}} [h_{i,j} * e_{i,j}]^2. \quad (3.12)$$

The probability that the estimated array $\hat{\mathbf{X}}_{\mathcal{E}}$ given by MLPD has greater likelihood than $\mathbf{X}_{\mathcal{E}}$ can be expressed as

$$\begin{aligned} \Pr(\mathbf{E}'') &= \int_{d^2(\mathbf{E})}^{+\infty} \frac{1}{\sqrt{2\pi \cdot 4\sigma^2 d^2(\mathbf{E})}} \cdot \exp\left(\frac{-\tau^2}{2 \cdot 4\sigma^2 d^2(\mathbf{E})}\right) d\tau \\ &= Q\left(\frac{d(\mathbf{E})}{2\sigma}\right). \end{aligned} \quad (3.13)$$

Note that $Q(\cdot)$ denotes the tail probability of the standard normal distribution.

Since $\mathbf{E}'$ and $\mathbf{E}''$ are independent of each other, the probability of the particular independent error event $\mathbf{E}$ is given by

$$\begin{aligned} \Pr(\mathbf{E}) &= \Pr(\mathbf{E}') \cdot \Pr(\mathbf{E}'') \\ &= Q\left(\frac{d(\mathbf{E})}{2\sigma}\right) \cdot \prod_{(i,j) \in \mathcal{E}} \frac{M_a - |e_{i,j}|}{M_a}. \end{aligned} \quad (3.14)$$

3.2.3 Probability of Symbol Error

We first denote E as the set of all possible independent error events $\mathbf{E}$ starting from the point (k_1, l_1) . Then the probability that any independent error event starts from the point (k_1, l_1) is bounded by

$$\Pr(E) \leq \sum_{\mathbf{E} \in E} \Pr(\mathbf{E}). \quad (3.15)$$

From the above expression, we observe that the probability of a union of events cannot be greater than the sum of their individual probabilities. If we denote D as the set of all possible $d(\mathbf{E})$ and for each $d \in D$, we denote E_d as the subset of error events for which $d(\mathbf{E}) = d$, then (3.15) can be rewritten as

$$\Pr(E) \leq \sum_{d \in D} Q\left(\frac{d}{2\sigma}\right) \sum_{\mathbf{E} \in E_d} \left(\prod_{(i,j) \in \mathcal{E}} \frac{M_a - |e_{i,j}|}{M_a} \right). \quad (3.16)$$

Note that the above expression is independent of (k_1, l_1) . It may be comprehended as the probability of an error event per unit time. Thus the probability of symbol error can be computed by weighting each error event $\mathbf{E}$ with the corresponding number of symbol errors $w_s(\mathbf{E})$, resulting in

$$\begin{aligned} \Pr(e) &\leq \sum_{\mathbf{E} \in E} w_s(\mathbf{E}) \Pr(\mathbf{E}) \\ &= \sum_{d \in D} Q\left(\frac{d}{2\sigma}\right) \sum_{\mathbf{E} \in E_d} w_s(\mathbf{E}) \left(\prod_{(i,j) \in \mathcal{E}} \frac{M_a - |e_{i,j}|}{M_a} \right) \end{aligned} \quad (3.17)$$

Note that in (3.17), as d increases, the term $Q(d/2\sigma)$ decreases exponentially which decreases much faster than the increase of sum of the rest terms $\sum_{\mathbf{E} \in E_d} w_s(\mathbf{E}) \left(\prod_{(i,j) \in \mathcal{E}} \frac{M_a - |e_{i,j}|}{M_a} \right)$. Thus, the whole term $Q(d/2\sigma) \sum_{\mathbf{E} \in E_d} w_s(\mathbf{E}) \left(\prod_{(i,j) \in \mathcal{E}} \frac{M_a - |e_{i,j}|}{M_a} \right)$ becomes very insignificant for large d . Therefore, the sum in the right hand side of (3.17) converges.

Since the Q-function decreases exponentially as the argument increases, (3.17) will be dominated by some terms with small $d/2\sigma$. In [95], the authors investigated the distance properties and search of error events. It has been shown that for the 2-D partial response class-1 (PR1) channel $\mathbf{H} = [1 \ 1; 1 \ 1]$, minimum distance error events

are of the form

$$\mathbf{E} = \begin{bmatrix} 1 & -1 & \dots \\ -1 & 1 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad (3.18)$$

or

$$\mathbf{E} = \begin{bmatrix} -1 & 1 & \dots \\ 1 & -1 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix} \quad (3.19)$$

which is composed of 1's and -1 's in an alternate order and their negatives. We further observe that for a generalized 2-D channel, error events with small distance also tend to be of the same patterns as (3.18) or (3.19). Thus, instead of calculating all the error events, we can well approximate the upper bound by just considering a fraction of error events in the form of (3.18) and (3.19), i.e.,

$$\begin{aligned} \Pr(e) &\leq \sum_{\mathbf{E} \in E} w_s(\mathbf{E}) \Pr(\mathbf{E}) \\ &\simeq \sum_{\mathbf{E} \in E_A} w_s(\mathbf{E}) \Pr(\mathbf{E}) \end{aligned} \quad (3.20)$$

where E_A is the set of error events in the form of (3.18) and (3.19). Note that at small signal-to-noise ratio (SNR) region, the BER of the upper bound given by (3.17) and (3.20) can be higher than 0.5 which is the lowest possible BER for equally likely bits 0 and 1. This is because the upper bounds are very loose for low SNR and tight only for high SNR.

Similar to the analysis for the 1-D ISI channel in [35], the lower bound for 2-D ISI

is defined as

$$\Pr(e) \geq K_0 Q\left(\frac{d_{min}}{2\sigma}\right) \quad (3.21)$$

where d_{min} is the minimum value of $d(\mathbf{E})$. Let $E_{d_{min}}$ be the set of error events $\mathbf{E}$ with error event distance d_{min} , K_0 is the probability that the input data $\mathbf{X}$ will be such that $\mathbf{X} + \mathbf{E}$ is an allowable input array for at least one $\mathbf{E} \in E_{d_{min}}$. Equation (3.21) can be proved similarly as that of the 1-D ISI case [34]. With probability K_0 , the input data array $\mathbf{X}$ will be such that there is a probability $Q(d_{min}/2\sigma)$ of confusing with some allowable data array $\mathbf{X} + \mathbf{E}$, even with a genie telling us to choose between $\mathbf{X}$ and $\mathbf{X} + \mathbf{E}$. Since the actual detector cannot do better than the genie-aided detector, the probability of symbol error is lower bounded by $K_0 Q(d_{min}/2\sigma)$. The minimum distance calculation of a certain class of 2-D interference channels can be found in [95] and [96].

3.3 Numerical Results and Discussions

In this section, we will present some numerical results based on the lower and upper bounds derived in the previous section. Here, we consider a symbol alphabet of $A = \{0, 1\}$. Thus, the probability of symbol error becomes probability of bit error. The bit-error rate (BER) results are plotted against SNR which is defined as

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum_{i=0}^{M_h-1} \sum_{j=0}^{N_h-1} (h_{i,j})^2}{2\sigma^2} \right) \text{ dB}. \quad (3.22)$$

We consider two channel response matrices given by [94]:

$$\mathbf{H}_1 = \begin{pmatrix} 0.0327 & 0.181 & 0.0327 \\ 0.181 & 1 & 0.181 \\ 0.0327 & 0.181 & 0.0327 \end{pmatrix} \quad (3.23)$$

and

$$\mathbf{H}_2 = \begin{pmatrix} 0.0993 & 0.352 & 0.0993 \\ 0.352 & 1 & 0.352 \\ 0.0993 & 0.352 & 0.0993 \end{pmatrix}. \quad (3.24)$$

We adopt these two channel response matrices based on two reasons. Firstly, they are exactly the same channel parameters used in [94] based on which we compare our results. Secondly, the channels $\mathbf{H}_1$ and $\mathbf{H}_2$ belong to two groups of channels whose MLPD lower and upper bounds converge and differ, respectively, at high SNR.

Since it is computationally prohibitive to consider all the error events up to infinite sizes and the contributions of error events with large sizes are very marginal as shown in equation (3.17), we follow the same way as in [94] to consider all ternary error arrays within a (4×4) region. The upper bound is calculated using equation (3.17) by taking into account all independent error events within the (4×4) arrays and the lower bound is calculated using equation (3.21).

The performance bounds of these two channels are plotted in Fig. 3.2. Since we use the same channels as [94], the performance of MLPD for the two channels is identical to that in [94]. This consistent result confirms the accuracy of our alternative way of analyzing the MLPD bound. In Fig. 3.2, we also present the approximated upper bound given in (3.20). The approximated upper bound method considers all the error

Figure 3.2. The upper and lower bounds of the MLPD BER performance.

patterns in the form of (3.18) and (3.19) within the (4×4) region. Since the computation of approximated upper bound consists of fewer error events, the approximated upper bound lies between the lower bound and upper bound. However, it can be observed from Fig. 3.2 that the approximated upper bound is nearly identical to the upper bound when BER is lower than 10^{-3} . For the exhaustive search within an $(N_r \times N_c)$ region, $3^{N_r N_c}$ error events need to be searched among which all the independent error events should be included in the computation of (3.17). This approach results in a very high computational complexity especially when N_r and N_c become large. However, in an $(N_r \times N_c)$ region, we can only construct $N_r \cdot N_c$ error events in the form of (3.18) or (3.19). For example, when $N_r = N_c = 2$, 3^4 error events need to be considered during the exhaustive search. But there are only 4 error events, namely,

Figure 3.3. Performance bound as a function of a and b at SNR=16 dB.

$\{+1\}$, $\{+1, -1\}$, $\{+1, -1\}^T$, and $\{+1, -1; -1, +1\}$, in the form of (3.18), and the corresponding 4 negative error events in the form of (3.19). Thus, we only need to consider $2N_r N_c$ error events in the computation of the approximated upper bound. This translates into a considerable amount of saving in computational complexity.

In Fig. 3.3, we show the upper bound on the BER with various channel interference levels. Here, we consider a class of separable channels with

$$\mathbf{H} = \begin{pmatrix} ab & b & ab \\ a & 1 & a \\ ab & b & ab \end{pmatrix}. \quad (3.25)$$

In Fig. 3.3, SNR is fixed to be 16 dB and four curves with various a and b are

Table 3.1. Six smallest values of $d/2\sigma$ and the corresponding error patterns for various a and b at SNR=16 dB.

	$d/2\sigma$, error pattern			
	$a = b = 0.2$	$a = b = 0.4$	$a = b = 0.65$	$a = b = 0.9$
1	4.4615, $\mathbf{E}_{1,1}$	3.5152, $\mathbf{E}_{2,2}$	2.6358, $\mathbf{E}_{2,2}$	2.7927, $\mathbf{E}_{2,2}$
2	5.0066, $\mathbf{E}_{1,2}$	3.5821, $\mathbf{E}_{3,2}$	2.7425, $\mathbf{E}_{3,2}$	3.2929, $\mathbf{E}_{2,3}$
3	5.0066, $\mathbf{E}_{2,1}$	3.5821, $\mathbf{E}_{2,3}$	2.7425, $\mathbf{E}_{2,3}$	3.2929, $\mathbf{E}_{3,2}$
4	5.6182, $\mathbf{E}_{2,2}$	3.6504, $\mathbf{E}_{3,3}$	2.8535, $\mathbf{E}_{3,3}$	3.5299, $\mathbf{E}_{1,2}$
5	5.6304, $\mathbf{E}_{1,3}$	3.9602, $\mathbf{E}_{1,2}$	3.4293, $\mathbf{E}_{1,2}$	3.5299, $\mathbf{E}_{2,1}$
6	5.6304, $\mathbf{E}_{3,1}$	3.9602, $\mathbf{E}_{2,1}$	3.4293, $\mathbf{E}_{2,1}$	3.8826, $\mathbf{E}_{3,3}$

plotted. It can be observed that in general, the system performance degrades as the interference levels a and b increase. However, this performance degradation is not monotonic. In a certain range, increasing the interference level can even improve the error performance. A similar result has also been shown in [97] in which the non-monotonic relationship between BER and ITI level was observed. This phenomenon, although contrary to our intuition, can be explained by taking a closer look at the spectrum of $d/2\sigma$. Table 3.1 lists six smallest values of $d/2\sigma$ and the corresponding error patterns for various combinations of a and b at SNR = 16 dB. Parts of the spectrums of $d/2\sigma$ for various a and b at SNR = 16 dB are shown in Fig. 3.4. It can be observed that when $a = b = 0.65$, there are more small $d/2\sigma$ terms than other combinations of a and b . As equation (3.17) is dominated by error events with small $d/2\sigma$, the performance becomes worse when a and b are around 0.65. In Table 3.1, $\mathbf{E}_{1,1} = \pm 1$, $\mathbf{E}_{1,2} = \pm[1, -1]$, $\mathbf{E}_{2,1} = \pm[1, -1]^T$, $\mathbf{E}_{1,3} = \pm[1, -1, 1]$, $\mathbf{E}_{3,1} = \pm[1, -1, 1]^T$, $\mathbf{E}_{2,2} = \pm[1, -1; -1, 1]$, $\mathbf{E}_{2,3} = \pm[1, -1, 1; -1, 1, -1]$, $\mathbf{E}_{3,2} = \pm[1, -1, 1; -1, 1, -1]^T$, and $\mathbf{E}_{3,3} = \pm[1, -1, 1; -1, 1, -1; 1, -1, 1]$. It can also be observed from Table 3.1 that the error patterns with the six smallest values of $d/2\sigma$ are all in the form of (3.18) or (3.19).

Figure 3.4. Parts of the spectrums of $d/2\sigma$ for various a and b at $\text{SNR}=16$ dB.

Thus, we conclude that (3.20) is a good approximation of the upper bound for BER. As the MLPD performance is dependent on the channel condition, we also observe that the approximated upper bound is dependent on the channel parameters.

In Fig. 3.5, we show the ratio of the approximated upper bound and the upper bound with various values of a and b . The SNR is fixed at $\text{SNR} = 16$ dB. It can be observed from Fig. 3.5 that in general, the approximated upper bound well approximates the upper bound. Similar to the previous case, we observe that the ratio of the approximated upper bound and the upper bound does not decrease monotonically with the parameters a and b . This is because when a and b are around 0.65, the error patterns in the form of (3.18) or (3.19) become less dominating in the calculation of

Figure 3.5. Ratio of the approximated upper bound and the upper bound for various a and b at SNR=16 dB.

(3.17) than other choices of a and b . Although the channel considered here is a special case, the same trend can also be observed for other channel conditions.

3.4 Conclusion

In this chapter, we have studied the MLPD for a 2-D interference channel corrupted by AWGN. We have derived the performance bound of MLPD in a new way. Numerical results based on the new performance bounding method have been compared to that of the traditional one and it has been shown that the proposed scheme gives accurate results. Furthermore, an approximated upper bounding method has been proposed which can accurately predict the upper bound with significant reduction in the compu-

tational complexity. A parametric effect of the channel interference level has also been investigated.

Chapter 4

Joint Iterative Detection/Decoding Scheme

In this chapter, we propose a joint iterative detection/decoding scheme (JIDDS) for two-dimensional (2-D) inter-symbol interference (ISI) channels with low-density parity-check (LDPC) codes. Similar to the iterative detection and decoding algorithm (IDDA) [84] and iterative row-column soft-decision feedback algorithm (IRCSDFA) [86],[87], our channel detector also consists of two constituent detectors that work iteratively to mitigate the 2-D ISI. There are essentially two major differences between our proposed algorithm and the existing IDDA and IRCSDFA algorithms. Firstly, our detection algorithm can be applied to a more generalized discrete 2-D ISI channel. The channel does not need to be separable as required in [84]. Secondly, in IRCSDFA, the two constituent detectors are concatenated in parallel, and they both access and process the readback signal and exchange weighted binary soft information. On the other hand, for our proposed detection algorithm, the two constituent detectors are serially concatenated. The down-track constituent detector accesses the readback signal and generates symbol-based soft information which is further processed in the cross-track

Figure 4.1. Read channel model for a 2-D ISI system.

constituent detector. The soft information exchanged between the two constituent detectors is not weighted. Furthermore, we also propose a reduced-complexity channel detector to reduce the system complexity and make the detection scheme more suitable for practical applications. Simulation results show that our proposed JIDDS outperforms some existing detection algorithms, such as the suboptimal strip-wise algorithm (SSWA, the so called MAP detection algorithm in [83]), the Markov chain Monte Carlo (MCMC)-based method [92], and the partial ITI estimation (PIE)-based algorithm [93]. It is further demonstrated that the proposed reduced-complexity channel detector can greatly reduce the system complexity at the expense of only very marginal performance loss.

4.1 System Model

As mentioned previously, we consider a more generalized discrete 2-D ISI channel. The corresponding channel response matrix of size $M_h \times N_h$ can be expressed as

$$\begin{aligned}
 \mathbf{H} &= [\mathbf{h}_0, \mathbf{h}_1, \dots, \mathbf{h}_{N_h-1}] \\
 &= \begin{bmatrix} h_{0,0} & h_{0,1} & \dots & h_{0,N_h-1} \\ h_{1,0} & h_{1,1} & \dots & h_{1,N_h-1} \\ \dots & \dots & & \dots \\ h_{M_h-1,0} & h_{M_h-1,1} & \dots & h_{M_h-1,N_h-1} \end{bmatrix}
 \end{aligned} \tag{4.1}$$

where each $\mathbf{h}_i$ is an M_h -tuple of real numbers. Note that we do not require the channel response matrix to be symmetric or separable.

In Fig. 4.1, we show the channel model of a 2-D ISI system. At the front end, an array of binary bits $\{u_i \in \{0, 1\}, i = 1, 2, \dots, K\}$ is first passed to the LDPC encoder. The one-dimensional (1-D) encoded array $\{s_i \in \{0, 1\}, i = 1, 2, \dots, N\}$ is first bipolar modulated to $c_i = 2s_i - 1$. It is then distributed on a 2-D array $\{x_{j,k} \in \{-1, 1\}, j = 1, 2, \dots, N_r, k = 1, 2, \dots, N_c\}$ with $N = N_r \times N_c$ according to a 1-D sequence to 2-D array mapping scheme. We refer to this process as an interleaving operation. The deinterleaver is a reverse process that converts a 2-D array $\{x_{j,k}, j = 1, 2, \dots, N_r, k = 1, 2, \dots, N_c\}$ back to 1-D sequence $\{c_i, i = 1, 2, \dots, N\}$. The values N_r and N_c represent the numbers of rows and columns of the 2-D array, respectively. We assume the 2-D array $\{x_{j,k} : j = 1, 2, \dots, N_r, k = 1, 2, \dots, N_c\}$ is surrounded by -1 s, i.e., $x_{j,k} = -1$ for $\{j \leq 0, j \geq N_r + 1\}$ or $\{k \leq 0, k \geq N_c + 1\}$. The readback signal, suffering from 2-D ISI and AWGN, can be expressed as

$$\begin{aligned} r_{i,j} &= y_{i,j} + v_{i,j} \\ &= \sum_{m=0}^{M_h-1} \sum_{n=0}^{N_h-1} h_{m,n} x_{i-m,j-n} + v_{i,j} \end{aligned} \quad (4.2)$$

where $y_{i,j}$ is the noiseless channel output and $v_{i,j}$ represents the AWGN component with zero mean and variance of σ^2 . If we denote the cross-track direction M_h -tuple coded bits at the i -th track time instant j as a symbol $\mathbf{x}_{i,j} = [x_{i,j}, x_{i-1,j}, \dots, x_{i-M_h+1,j}]^T$, (4.2) can be expressed in another way as

$$r_{i,j} = \sum_{n=0}^{N_h-1} \mathbf{x}_{i,j-n}^T \cdot \mathbf{h}_n + v_{i,j}. \quad (4.3)$$

Figure 4.2 shows a factor graph representation of the system. In this figure, we set

Figure 4.2. Factor graph representation of the system.

$M_h = N_h = 3$ and each variable node is checked by 2 check nodes and each check node is connected to 3 variable nodes. Each grid represents a recorded bit which corresponds uniquely to a variable node of the LDPC code. The shaded circles represent some check nodes and the shaded grids represent the corresponding variable nodes connected to those check nodes. The grids with crosses represent all the binary signals that can interfere with $x_{i,j}$ and they form the interference region of $x_{i,j}$. Similarly, $x_{i,j}$ can also interfere with all the bits within its interference region.

4.2 Detection/Decoding Algorithm

In this section, we describe the 2-D iterative detection algorithm with LDPC codes. In Fig. 4.3, we show the system block diagram of the proposed detector and decoder. In

particular, we refer to the block which mitigates the 2-D ISI as detector and the block which decodes the LDPC codes as decoder as shown in Fig. 4.3. For the detection of 2-D ISI channels, it might be intuitive to first detect the column-wise symbols which span multiple tracks and then use the symbol information to detect the bits on the center track. This is basically the idea adopted in SSWA [83]. However, our proposed channel detector does not work this way. The proposed iterative detection scheme is shown in Fig. 4.4. As shown in Fig. 4.4, the proposed detector consists of two parts, namely, the down-track detector and cross-track detector. The down-track detector is operated on each track to provide the soft information of each symbol. Then instead of only detecting the bits on the center track, the cross-track detector makes use of the information of the symbols from the same column and detects the bits on the whole column. In other words, in our proposed JIDDS, we make use of the symbol information from many tracks jointly instead of individually as in SSWA. This joint-track detection scheme can provide superior performance comparing to the SSWA. We let the readback signals be first processed by the down-track detector in the down-track direction and then by the cross-track detector in the cross-track direction. This is simply because the MR head reads the recorded signals track by track in the down-track direction. In order to improve the system performance, we will introduce iterations between the two constituent detectors as well as iterations among those two detectors with the LDPC decoder as depicted in Figs. 4.3 and 4.4. In the following, we describe in details the proposed JIDDS algorithm.

4.2.1 Down-Track Detector

The objective of the down-track detector is to detect each symbol $\mathbf{x}_{i,j}$. In order to facilitate information exchange between the two constituent detectors, the down-track

Figure 4.5. Trellis structure of the down-track detector for $M_h = N_h = 3$.

are 64 states at a time instant with 8 paths entering and leaving each state. Based on the proposed trellis structure, we can calculate the coefficients of the log-domain symbol-based BCJR detection as

$$\gamma(S_{i,k}, S_{i,k+1}) = L_{id}(\mathbf{x}_{i,k}) - \frac{\left| r_{i,k} - \sum_{n=0}^{N_h-1} \mathbf{x}_{i,k-n}^T \cdot \mathbf{h}_n \right|^2}{2\sigma^2} \quad (4.4)$$

$$\alpha(S_{i,k+1}) = \max_{S_{i,k}}^* \{\alpha(S_{i,k}) + \gamma(S_{i,k}, S_{i,k+1})\} \quad (4.5)$$

$$\beta(S_{i,k}) = \max_{S_{i,k+1}}^* \{\beta(S_{i,k+1}) + \gamma(S_{i,k}, S_{i,k+1})\} \quad (4.6)$$

$$\begin{aligned} L_{od}(\mathbf{x}_{i,k}) &= \log(P(\mathbf{x}_{i,k} = U_t | \mathbf{r})) \\ &= \max_{U_t}^* \{\alpha(S_{i,k}) + \gamma(S_{i,k}, S_{i,k+1}) + \beta(S_{i,k+1})\} \end{aligned} \quad (4.7)$$

where $L_{id}(\mathbf{x}_{i,k})$ is the logarithmic *a priori* probability of the symbol $\mathbf{x}_{i,k}$, the subscript implies $L_{id}(\mathbf{x}_{i,k})$ is the input to the down-track detector. The parameter U_t represents the t -th element from the symbol alphabet U of size 2^{M_h} . For example, in Fig. 4.5, the first element is defined as $U_1 = [-1, -1, -1]^T$ and the first state becomes $[U_1, U_1]$. Furthermore, $L_{od}(\mathbf{x}_{i,k})$ denotes the generated *a posteriori* logarithmic probability of the symbol $\mathbf{x}_{i,k}$, and the subscript implies it is the output of the down-track detector. The variables $\alpha(S_{i,k})$ and $\beta(S_{i,k})$ are the forward and backward state metrics, respectively. They are initialized as

$$\alpha(S_{i,0}) = \begin{cases} 0, & \text{if } S_{i,0} = [U_1, U_1, \dots, U_1] \\ -\infty, & \text{other } S_{i,0} \end{cases} \quad (4.8)$$

and

$$\beta(S_{i,N_c+N_h-1}) = \begin{cases} 0, & \text{if } S_{i,N_c+N_h-1} = [U_1, U_1, \dots, U_1] \\ -\infty, & \text{other } S_{i,N_c+N_h-1} \end{cases} . \quad (4.9)$$

The above initializations are based on the assumption that the 2-D data array is sur-

rended by -1 s. The function $\max^*$ is defined as [98]

$$\begin{aligned}\max^*(x, y) &\triangleq \log(e^x + e^y) \\ &= \max(x, y) + \log(1 + e^{-|x-y|}).\end{aligned}\tag{4.10}$$

For more than two variables, the $\max^*$ function can be calculated in a recursive way as

$$\begin{aligned}\max^*(x, y, z) &\triangleq \log(e^x + e^y + e^z) \\ &= \max^*(\max^*(x, y), z).\end{aligned}\tag{4.11}$$

Note that the correction term $\log(1 + e^{-|x-y|})$ in (4.10) is small for most circumstances. Thus, $\max$ is a good approximation of $\max^*$.

Due to the symmetries of the channel response matrix, some of the symbols may have equal logarithmic probabilities and cannot be distinguished. For example, when $h_{0,N_h-1} = h_{M_h-1,N_h-1}$, symbols $[-1, \dots, 1]$ and $[1, \dots, -1]$ have the same logarithmic probability value and cannot be distinguished. Thus, we need to transfer the symbol-based information of the down-track detector to the cross-track detector for further detection. In the next subsection, we will describe the operation of the cross-track detector.

4.2.2 Cross-Track Detector

The objective of the cross-track detector is to generate the soft estimation of the corresponding binary bits. The inputs to the cross-track detector are the logarithmic probabilities of the symbols from each track. As mentioned in the previous section, the down-track detector cannot fully distinguish all the 2^{M_h} symbols in the down-track direction detection. Thus it is necessary to adopt this cross-track detector that works

Figure 4.6. Trellis structure of the cross-track detector for $M_h = 3$ and $N_r = 5$.

in another direction to detect the desired binary data.

In order to facilitate the derivation, we concentrate on the j -th column, i.e., we want to detect the binary data of $x_{i,j}$, for $i = 1, 2, \dots, N_r$. The algorithm in the cross-track detector is also a trellis-based algorithm. A state at time unit i in the trellis is defined as the previous $M_h - 1$ bits in the cross-track direction, i.e., $s_{i,j} = [x_{i-1,j}, x_{i-2,j}, \dots, x_{i-M_h,j}]^T$. We use the lower case s to distinguish the trellis state in the cross-track detector from that in the down-track detector. The number of states in the trellis of cross-track detector at each time unit is 2^{M_h-1} . Since the states of adjacent time units are shifted by one bit, there are two paths entering and leaving each state. Note that as our input binary data are surrounded by -1 s at the top and bottom, the initial and terminating states in the trellis should be both $[-1, -1, \dots, -1]^T$. In Fig. 4.6, we show the trellis structure used in the cross-track direction detection with parameters of $M_h = 3$ and $N_c = 5$.

We denote $\mathbb{L}^j = \{\mathbb{L}_i^j, i = 1, \dots, N_c + M - 1\}$ where $\mathbb{L}_i^j = \log P(\mathbf{x}_{i,j} | \mathbf{r}) = L_{ic}(\mathbf{x}_{i,j})$ is the input logarithmic probability to the cross-track detector. The goal of the soft-output cross-track detector is to calculate the logarithmic likelihood ratio (LLR) of each bit,

given the logarithmic probabilities of each symbols $\mathbb{L}^j$. This LLR can be expressed as

$$L_{oc}(x_{i,j}) = \log \left(\frac{P(x_{i,j} = 1 | \mathbb{L}^j)}{P(x_{i,j} = -1 | \mathbb{L}^j)} \right). \quad (4.12)$$

The subscript of $L_{oc}(x_{i,j})$ implies it is the output of the cross-track detector. In order to incorporate the trellis into the computations, we rewrite $L_{oc}(x_{i,j})$ as

$$L_{oc}(x_{i,j}) = \log \frac{\sum_{V_1} P(s_{i,j}, s_{i+1,j}, \mathbb{L}^j)}{\sum_{V_{-1}} P(s_{i,j}, s_{i+1,j}, \mathbb{L}^j)} \quad (4.13)$$

where V_1 is the set of pairs $(s_{i,j}, s_{i+1,j})$ that correspond to the event $x_{i,j} = 1$. For example, in the trellis shown in Fig. 4.6, V_1 corresponds to the set $\{([-1; -1], [1; -1]), ([-1; 1], [1; -1]), ([1; -1], [1; 1]), ([1; 1], [1; 1])\}$. The set V_{-1} is similarly defined. To calculate $L_{oc}(x_{i,j})$, we first introduce the following expressions based on the BCJR algorithm [33]

$$\begin{aligned} & P(s_{i,j}, s_{i+1,j}, \mathbb{L}^j) \\ &= P(s_{i,j}, s_{i+1,j}, \mathbb{L}_{t < i}^j, \mathbb{L}_i^j, \mathbb{L}_{t > i}^j) \\ &= P(\mathbb{L}_{t > i}^j | s_{i,j}, s_{i+1,j}, \mathbb{L}_{t < i}^j, \mathbb{L}_i^j) \cdot P(s_{i,j}, s_{i+1,j}, \mathbb{L}_{t < i}^j, \mathbb{L}_i^j) \\ &= P(\mathbb{L}_{t > i}^j | s_{i+1,j}) \cdot P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}, \mathbb{L}_{t < i}^j) \cdot P(s_{i,j}, \mathbb{L}_{t < i}^j) \\ &= P(\mathbb{L}_{t > i}^j | s_{i+1,j}) \cdot P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}) \cdot P(s_{i,j}, \mathbb{L}_{t < i}^j) \\ &= \mathbf{b}(s_{i+1,j}) \cdot \mathbf{c}(s_{i,j}, s_{i+1,j}) \cdot \mathbf{a}(s_{i,j}) \end{aligned} \quad (4.14)$$

where $\mathbf{a}(s_{i,j}) = P(s_{i,j}, \mathbb{L}_{t < i}^j)$, $\mathbf{b}(s_{i+1,j}) = P(\mathbb{L}_{t > i}^j | s_{i+1,j})$ and $\mathbf{c}(s_{i,j}, s_{i+1,j}) = P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j})$ are the forward metric, backward metric and branch metric, respectively. The forward

metric and backward metric can be calculated via [33]

$$\begin{aligned}
\mathbf{a}(s_{i+1,j}) &\triangleq P(s_{i+1,j}, \mathbb{L}_{t < i+1}^j) \\
&= \sum_{s_{i,j}} P(s_{i,j}, s_{i+1,j}, \mathbb{L}_{t < i+1}^j) \\
&= \sum_{s_{i,j}} P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}, \mathbb{L}_{t < i}^j) P(s_{i,j}, \mathbb{L}_{t < i}^j) \\
&= \sum_{s_{i,j}} P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}) P(s_{i,j}, \mathbb{L}_{t < i}^j) \\
&= \sum_{s_{i,j}} \mathbf{c}(s_{i,j}, s_{i+1,j}) \mathbf{a}(s_{i,j})
\end{aligned} \tag{4.15}$$

and

$$\begin{aligned}
\mathbf{b}(s_{i,j}) &\triangleq P(\mathbb{L}_{t > i-1}^j | s_{i,j}) \\
&= \sum_{s_{i+1,j}} P(\mathbb{L}_{t > i-1}^j, s_{i+1,j} | s_{i,j}) \\
&= \sum_{s_{i+1,j}} P(\mathbb{L}_i^j, \mathbb{L}_{t > i}^j, s_{i+1,j}, s_{i,j}) / P(s_{i,j}) \\
&= \sum_{s_{i+1,j}} P(\mathbb{L}_{t > i}^j | \mathbb{L}_i^j, s_{i+1,j}, s_{i,j}) P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}) \\
&= \sum_{s_{i+1,j}} P(\mathbb{L}_{t > i}^j | s_{i+1,j}) P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}) \\
&= \sum_{s_{i,j}} \mathbf{b}(s_{i+1,j}) \mathbf{c}(s_{i,j}, s_{i+1,j}).
\end{aligned} \tag{4.16}$$

Since the trellis originates from and terminates at the same state $[-1, -1, \dots, -1]^T$, the forward and backward metrics should be initialized using the followings:

$$\mathbf{a}(s_{0,j}) = \begin{cases} 1, & \text{if } s_{0,j} = [-1, -1, \dots, -1]^T \\ 0, & \text{other } s_{0,j} \end{cases} \tag{4.17}$$

and

$$\mathfrak{b}(s_{N_r+M_h-1,j}) = \begin{cases} 1, & \text{if } s_{N_r+M_h-1,j} = [-1, -1, \dots, -1]^T \\ 0, & \text{other } s_{N_r+M_h-1,j} \end{cases}. \quad (4.18)$$

The branch metric can be expressed as

$$\begin{aligned} \mathfrak{c}(s_{i,j}, s_{i+1,j}) &\triangleq P(s_{i+1,j}, \mathbb{L}_i^j | s_{i,j}) \\ &= \frac{P(s_{i,j}, s_{i+1,j})}{P(s_{i,j})} \cdot \frac{P(s_{i,j}, s_{i+1,j}, \mathbb{L}_i^j)}{P(s_{i,j}, s_{i+1,j})} \\ &= P(s_{i+1,j} | s_{i,j}) P(\mathbb{L}_i^j | s_{i,j}, s_{i+1,j}) \\ &= P(x_{i,j}) P(\mathbb{L}_i^j | s_{i,j}, s_{i+1,j}) \end{aligned} \quad (4.19)$$

where $P(x_{i,j})$ is the *a priori* probability of the bit $x_{i,j}$ and it corresponds to the event $s_{i,j} \rightarrow s_{i+1,j}$, and $P(\mathbb{L}_i^j | s_{i,j}, s_{i+1,j}) = P(\mathbf{x}_{i,j})$ which can be obtained from the down-track detector.

If we denote the probability variables with tildes as the corresponding logarithmic probabilities, the log-domain cross-track detector can be implemented as

$$\tilde{\mathfrak{c}}(s_{i,j}, s_{i+1,j}) = \mathbb{L}_i^j + \log(P(x_{i,j})) \quad (4.20)$$

$$\tilde{\mathfrak{a}}(s_{i+1,j}) = \max_{s_{i,j}}^* \{ \tilde{\mathfrak{a}}(s_{i,j}) + \tilde{\mathfrak{c}}(s_{i,j}, s_{i+1,j}) \} \quad (4.21)$$

$$\tilde{\mathfrak{b}}(s_{i,j}) = \max_{s_{i+1,j}}^* \{ \tilde{\mathfrak{b}}(s_{i+1,j}) + \tilde{\mathfrak{c}}(s_{i,j}, s_{i+1,j}) \} \quad (4.22)$$

$$\begin{aligned}
L_{oc}(x_{i,j}) &= \max_{V_1}^* \{ \tilde{\mathbf{a}}(s_{i,j}) + \tilde{\mathbf{c}}(s_{i,j}, s_{i+1,j}) + \tilde{\mathbf{b}}(s_{i+1,j}) \} \\
&\quad - \max_{V_{-1}}^* \{ \tilde{\mathbf{a}}(s_{i,j}) + \tilde{\mathbf{c}}(s_{i,j}, s_{i+1,j}) + \tilde{\mathbf{b}}(s_{i+1,j}) \}.
\end{aligned} \tag{4.23}$$

4.2.3 Extrinsic Information Transfer

The proposed JIDDS mainly consists of three blocks, namely, the down-track detector, the cross-track detector and the LDPC decoder as shown in Figs. 4.3 and 4.4. To achieve better system performance, it is desirable to introduce iterations among the three blocks. We can introduce four iterations in the joint detection/decoding system, namely, decoder iteration, detector iteration, inner iteration and outer iteration which represent the iteration inside the LDPC decoder, iteration between down-track detector and cross-track detector, iteration between cross-track detector and LDPC decoder, and lastly the outer iteration among down-track detector, cross-track detector and LDPC decoder, respectively. We denote the numbers of iterations in the decoder iteration, detector iteration, inner iteration and outer iteration as I_{dec} , I_{det} , I_{in} and I_{out} , respectively. Following the message-passing algorithm explained in Chapter 2, the information transferred between functions or blocks should be extrinsic. Next, we will explain the detailed extrinsic information transfer within the four types of iterations.

Since the cross-track detector and LDPC decoder both work on binary field, the binary extrinsic information between the cross-track detector and the LDPC decoder can be conveniently obtained by subtracting the *a priori* LLRs from the *a posteriori* LLRs. Let the deinterleaved *a posteriori* LLRs from the cross-track detector be denoted as $\{L_{oc}(c_n), n = 1, \dots, N\}$ and the *a posteriori* LLRs from the output of LDPC decoder be denoted as $\{L_{ol}(c_n), n = 1, \dots, N\}$, then we have

$$L_{ext}^1(c_n) = L_{oc}(c_n) - L_{ext}^2(c_n) \tag{4.24}$$

and

$$L_{ext}^2(c_n) = L_{ol}(c_n) - L_{ext}^1(c_n) \quad (4.25)$$

where $L_{ext}^1(c_n)$ and $L_{ext}^2(c_n)$ represent the extrinsic LLRs which are input to the LDPC decoder and feedback to the channel detector, respectively.

The LDPC decoder processes the input information $\{L_{ext}^1(c_n), n = 1, \dots, N\}$ and generate *a posteriori* LLRs $\{L_{ol}(c_n), n = 1, \dots, N\}$. The iterative calculations in the LDPC decoder mainly consist of the variable node update and the check node update [19]. The LLR value z_{mn} of the n -th bit which is sent from a variable node n to a check node m can be calculated as

$$z_{mn} = L_{ext}^1(c_n) + \sum_{j \in D_v(n) \setminus m} q_{jn} \quad (4.26)$$

where $D_v(n) \setminus m$ represents all the check nodes connected to the variable node n except for the check node m . Note that q_{mn} is the LLR of bit n which is sent from the check node m to the variable node n . The update of q_{mn} can be calculated as

$$q_{mn} = 2 \tanh^{-1} \left\{ \prod_{k \in D_c(m) \setminus n} \tanh[z_{mk}/2] \right\} \quad (4.27)$$

where $D_c(m) \setminus n$ represents all the variable nodes connected to the check node m except for the variable node n . The functions $\tanh(x)$ and $\tanh^{-1}(x)$ are defined as

$$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}. \quad (4.28)$$

and

$$\tanh^{-1}(x) = \frac{1}{2} \ln \left(\frac{1+x}{1-x} \right). \quad (4.29)$$

The above variable node update and the check node update are repeated for I_{dec} times. Finally, the output LLRs of the LDPC decoder can be calculated via

$$L_{ol}(c_n) = L_{ext}^1(c_n) + \sum_{j \in D_v(n)} q_{jn} \quad (4.30)$$

where $D_v(n)$ represents all the check nodes connected to the variable node n . Note that the above LDPC decoding algorithm is equivalent to that explained in Chapter 2. The only difference is that here we perform all the calculations in the form of LLRs instead of probabilities as in Chapter 2.

The extrinsic logarithmic probabilities transferred from the cross-track detector to the down-track detector can be obtained via

$$L_{oc}(\mathbf{x}_{k,j}) = \tilde{\mathbf{a}}(s_{k,j}) + \log(P(x_{k,j})) + \tilde{\mathbf{b}}(s_{k+1,j}). \quad (4.31)$$

Note that the term $\tilde{\mathbf{a}}(s_{k,j}) + \tilde{\mathbf{c}}(s_{k,j}, s_{k+1,j}) + \tilde{\mathbf{b}}(s_{k+1,j})$ is the logarithmic *a posteriori* probability of the symbol $\mathbf{x}_{k,j}$. Thus, (4.31) is equivalent to excluding the *a priori* information $\mathbb{L}_k^j$ from the term $\tilde{\mathbf{c}}(s_{k,j}, s_{k+1,j})$ in the above expression. Since $L_{oc}(\mathbf{x}_{k,j})$ is already the extrinsic information, there is no need to exclude the *a priori* information from $L_{oc}(\mathbf{x}_{k,j})$ when it is feedback to the down-track detector.

The LDPC decoder does not work on the trellis structure and cannot generate symbol-based information directly as the cross-track detector. To feedback the information from the LDPC decoder to the down-track detector, we need to convert the

bit-based information to symbol-based information. Since the bits in a symbol are assumed to be independent of each other, the bit-to-symbol converter can be conveniently implemented via

$$L_{id}(\mathbf{x}_{k,j}) = \sum_{i=0}^{M_h-1} \log P(x_{k-i,j}) \quad (4.32)$$

where $\log P(x_{i,j} = -1) = -\max^*(0, L_{ext}^2(x_{i,j}))$ and $\log P(x_{i,j} = 1) = L_{ext}^2(x_{i,j}) + \log P(x_{i,j} = -1)$. Following that, $L_{oc}(\mathbf{x}_{k,j})$ in (4.31) and $L_{id}(\mathbf{x}_{k,j})$ in (4.32) are then used to update the *a priori* symbol-based logarithmic probability $L_{id}(\mathbf{x}_{i,k})$ in (4.4) in the next iteration. The extrinsic symbol-based log probability transferred from the down-track detector to the cross-track detector can be obtained by normalizing the remainder after subtracting the *a priori* log probability from the *a posteriori* log probability, i.e.,

$$L_{ic}(\mathbf{x}_{k,j}) = K_{k,j} (L_{od}(\mathbf{x}_{k,j}) - L_{id}(\mathbf{x}_{k,j})) \quad (4.33)$$

where $K_{k,j}$ is a normalization factor to make sure the sum of probabilities equals to 1.

The down-track detector and the cross-track detector work in the down-track direction and cross-track direction iteratively to generate LLR of each bit. Based on the soft information from the channel detector and the variable-check node constraints, the LDPC decoder refreshes the bit information and produces more reliable estimation of each bit. When transferring extrinsic information between the channel detector and LDPC decoder, the correlation of the LDPC coded array needs to be taken into consideration. Firstly, from the channel detector side, the algorithm in the channel detector is based on the assumption that all the binary bits within an interference region are independent of each other. In the process of feeding back the extrinsic information from the LDPC decoder to the channel detector, we should avoid the case where variable nodes

checked by the same check node lying within one interference region. Secondly, from the LDPC decoder side, the soft information of $x_{i,j}$ transferred from channel detector to the LDPC decoder is directly dependent on the information of bits from the interference region of $x_{i,j}$. If any pairs of bits from one interference region are checked by the same check node, a short cycle will be created, which will be harmful to the iterative decoding. Therefore, an interleaver is necessary for the joint detection/decoding system to make the LDPC coded 1-D sequence properly distributed on the 2-D array. If the parity check matrix of the LDPC code is constructed using the semi-random algorithm as proposed in [19], a block interleaver can generally work well. A block interleaver is a mapping scheme that distributes the 1-D sequence $\{c_i, i = 1, 2, \dots, N\}$ to the 2-D array $\{x_{j,k}, j = 1, 2, \dots, N_r, k = 1, 2, \dots, N_c\}$ such that $x_{j,k} = c_i$ with $j = \lfloor i/N_c \rfloor$ and $k = i - (j - 1) \times N_c$.

4.2.4 Reduced-Complexity Detection

In Table 4.1, we compare the computational complexities of each individual block of the proposed JIDDS scheme. We denote C_w and R_w as the column-weight and row-weight of the parity check matrix of the LDPC code, respectively. We also assume that the log operation in (4.10) can be achieved by a table look-up. It can be observed from Table 4.1 that the computational complexity of the whole system is dominated by the down-track detector. In order to further reduce the computational cost of the down-track detector, we apply the reduced-state BCJR algorithm [99],[100] to the detection in the down-track detector.

We define a state in the reduced-state trellis as $\hat{S}_{i,k} = [\mathbf{x}_{i,k-Q+1}, \dots, \mathbf{x}_{i,k-1}]$, with $Q \leq N_h$. In contrast to the full-state trellis in which a state is defined as $S_{i,k} = [\mathbf{x}_{i,k-N_h+1}, \dots, \mathbf{x}_{i,k-1}]$, the reduced state only consists of $Q - 1$ symbols instead of $N_h - 1$

Figure 4.7. Trellis structure of the reduced-state down-track detector for $N_h = 3$ and $Q = 2$.

symbols. Thus, the number of states is reduced from $2^{M_h(N_h-1)}$ to $2^{M_h(Q-1)}$, which greatly simplifies the trellis structure. In Fig. 4.7, we show a reduced-state trellis structure of the down-track detector with the parameters $M_h = 3$ and $Q = 2$. In this figure, at a time instant, the number of state becomes $2^3 = 8$, which is 8 times smaller than that in Fig. 4.5. The number of incoming and outgoing paths to each state is still equal to $2^{M_h} = 8$ in this case. It can be seen that the overall reduced-state

Table 4.1. Comparison of computational complexity per bit of each individual block

Block	Additions	Multiplications	Max	Table look-up
LDPC decoder	$C_w(5R_w - 8)$	-	$2C_w(R_w - 2)$	$2C_w(R_w - 2)$
Full-state down-track det	$11 \cdot 2^{M_h N_h}$	$2^{M_h N_h + 1}$	$3 \cdot 2^{M_h N_h}$	$2^{M_h N_h + 2}$
Reduced-state down-track det	$3 \cdot 2^{M_h Q + 2}$	$2^{M_h Q + 1}$	$5 \cdot 2^{M_h Q}$	$5 \cdot 2^{M_h Q}$
Cross-track det	$15 \cdot 2^{M_h}$	-	$4 \cdot 2^{M_h}$	$4 \cdot 2^{M_h}$
MCMC	$F_c(M_h + 1)^2(N_h + 1)^2$	$2F_c(M_h + 1)(N_h + 1)$	$2N_s$	$2N_s$
SSWA	$112^{N_h + 2} 3^{N_h - 1} + 16$	$2^{N_h + 2} 3^{N_h - 1}$	$2^{N_h + 2} 3^{N_h} + 8$	$2^{N_h + 5} 3^{N_h - 1} + 8$
Equalizer-based	$2l + 40$	$2l + 65$	8	32

trellis structure is much simpler than that for a full-state. Since the number of states is reduced, $y_{i,k}$ cannot be uniquely determined by a pair of states $(\hat{S}_{i,k}, \hat{S}_{i,k+1})$ and the channel response matrix $\mathbf{H}$. Thus, we need to define an estimated $\hat{y}_{i,k}$, given by

$$\hat{y}_{i,k} = \sum_{n=0}^{Q-1} \mathbf{x}_{i,k-n}^T \cdot \mathbf{h}_n + \sum_{n=Q}^{N_h-1} \hat{\mathbf{x}}_{i,k-n}^T \cdot \mathbf{h}_n. \quad (4.34)$$

From the above expression, we observe that $\hat{y}_{i,k}$ consists of two components. The first component $\sum_{n=0}^{Q-1} \mathbf{x}_{i,k-n}^T \cdot \mathbf{h}_n$ is directly associated with the pair of states $(\hat{S}_{i,k}, \hat{S}_{i,k+1})$. The second component $\sum_{n=Q}^{N_h-1} \hat{\mathbf{x}}_{i,k-n}^T \cdot \mathbf{h}_n$, however, cannot be determined from the reduced-state trellis. The symbol $\hat{\mathbf{x}}_{i,j-n}$, for $n = Q, \dots, N_h - 1$, denotes the survivor symbols that are associated with the state $\hat{S}_{i,k}$. For the i -th track, the *a posteriori* logarithmic probabilities can be calculated as

$$\hat{\gamma}(\hat{S}_{i,k}, \hat{S}_{i,k+1}) = L_{id}(\mathbf{x}_{i,k}) - \frac{|r_{i,k} - \hat{y}_{i,k}|^2}{2\sigma^2} \quad (4.35)$$

$$\hat{\alpha}(\hat{S}_{i,k+1}) = \max_{\hat{S}_{i,k}}^* \left\{ \hat{\alpha}(\hat{S}_{i,k}) + \hat{\gamma}(\hat{S}_{i,k}, \hat{S}_{i,k+1}) \right\} \quad (4.36)$$

$$\hat{\beta}(\hat{S}_{i,k}) = \max_{\hat{S}_{i,k+1}}^* \left\{ \hat{\beta}(\hat{S}_{i,k+1}) + \hat{\gamma}(\hat{S}_{i,k}, \hat{S}_{i,k+1}) \right\} \quad (4.37)$$

$$\begin{aligned} L(\mathbf{x}_{i,k}) &= \log(P(\mathbf{x}_{i,k} = U_t | r)) \\ &= \max_{U_t}^* \left\{ \hat{\alpha}(\hat{S}_{i,k}) + \hat{\gamma}(\hat{S}_{i,k}, \hat{S}_{i,k+1}) + \hat{\beta}(\hat{S}_{i,k+1}) \right\}. \end{aligned} \quad (4.38)$$

The above steps are very similar to those full-state calculations in (4.4)-(4.7). The only difference is that we need to determine the survivor symbols during the forward recursion. In the calculation of the forward recursion in (4.36), for each state $\hat{S}_{i,k+1}$, we compare all the incoming forward state metrics summed up with the corresponding trellis metrics and store the one with the largest value. For example, we want to find the survivor symbols associated with the state $\hat{S}_{i,k+1} = [U_1, U_1, \dots, U_1]$ at time instant $k+1$. There are 2^{M_h} states $\hat{S}_{i,k}$ at time instant k that are connected to $\hat{S}_{i,k+1}$. They are in the form of $\{[U_i, U_1, \dots, U_1], i = 1, 2, \dots, 2^{M_h}\}$. For all these 2^{M_h} incoming states, we compare their values $\hat{\alpha}(\hat{S}_{i,k}) + \hat{\gamma}(\hat{S}_{i,k}, \hat{S}_{i,k+1})$. The state with the largest value of $\hat{\alpha}(\hat{S}_{i,k}) + \hat{\gamma}(\hat{S}_{i,k}, \hat{S}_{i,k+1})$ is called the survivor state associated with $\hat{S}_{i,k+1}$. Suppose the survivor state is $\hat{S}_{i,k} = [U_t, U_1, \dots, U_1]$, then the survivor symbol associated with the state $\hat{S}_{i,k+1} = [U_1, U_1, \dots, U_1]$ at time instant $k+1$ is U_t . At time instant $k+2$, suppose $\hat{S}_{i,k+1} = [U_1, U_1, \dots, U_1]$ is a survivor state of a state $\hat{S}_{i,k+2}$, then the survivor symbol associated with the state $\hat{S}_{i,k+2}$ at time instant $k+2$ is U_1 . The previous two survivor symbols associated with the state $\hat{S}_{i,k+2}$ is now $[U_t, U_1]$. As the forward recursion proceeds, the survivor symbols form the survivor paths associated with each state. When we need to make use of the survivor symbols $\hat{\mathbf{x}}_{i,k-n}$ associated with a

certain state $\hat{S}_{i,k}$ in (4.34), we only need to search in the survivor path of the state $\hat{S}_{i,k}$.

From Table 4.1, we observe that the computational complexity of the reduced-state down-track detector is reduced by a factor $2^{M_h(N_h-Q)}$ as compared to that of the full-complexity case. The computational complexity comparison of our proposed detection algorithm and some other detection methods will be discussed in the next section.

4.2.5 Summary of the JIDDS

We assume that the simplest stopping criterion is adopted in our proposed JIDDS, i.e., the iterations are conducted for a pre-set number of times. The detection/decoding algorithm in the joint channel detection and LDPC codes decoding can be summarized as follows:

- Initialization
 1. Receive all the channel output $\{r_{i,j}, i = 1, \dots, N_r, j = 1, \dots, N_c\}$;
 2. Set all the *a priori* and extrinsic information $L_{id}(\mathbf{x}_{i,j})$, $L_{oc}(\mathbf{x}_{i,j})$, $L_{ext}^2(x_{i,j})$, $L_{ext}^2(c_n)$, and q_{mn} to zeros.
- Joint iterative detection/decoding
 1. Execute the full-state down-track detection algorithm as (4.4)-(4.7) or the reduced-state down-track detection algorithm as (4.34)-(4.38) to generate the log probabilities $L_{od}(\mathbf{x}_{i,j})$;
 2. Obtain the extrinsic log probabilities $L_{ic}(\mathbf{x}_{i,j})$ as (4.33);
 3. Execute the cross-track detection algorithm as (4.20)-(4.23) to obtain the LLRs $L_{oc}(x_{i,j})$ and compute the extrinsic log probabilities $L_{oc}(\mathbf{x}_{i,j})$ as (4.31) to use it update $L_{id}(\mathbf{x}_{i,j})$;

4. Repeat steps 1 to 3 for I_{det} times;
5. Deinterleave the LLRs $L_{oc}(x_{i,j})$ and obtain the extrinsic information $L_{ext}^1(c_n)$ as (4.24);
6. Compute the variable-to-check messages z_{mn} and check-to-variable messages q_{mn} iteratively as (4.26)-(4.27) for I_{dec} times and obtain the output LLRs $L_{ol}(c_n)$ of LDPC decoder as (4.30);
7. Compute the extrinsic information $L_{ext}^2(c_n)$ as (4.25). Interleave the information $L_{ext}^2(c_n)$ and use it to update the *a priori* information $\log(P(x_{i,j}))$ in the cross-track detector;
8. Repeat steps 3, 5, 6, and 7 for I_{in} times;
9. Obtain the extrinsic log probabilities $L_{ic}(\mathbf{x}_{i,j})$ as (4.32) and use it to update the *a priori* symbol-based information.
10. Repeat steps 1 to 9 for I_{out} times.

- Decode

1. Compute the estimated bits using $\hat{s}(n) = (\text{sign}(L_{ol}(c_n)) + 1) / 2$.

4.3 Simulation Results and Discussions

In this section, we investigate the effects of various parameters on the system performance. We also compare the performance and the computational complexity of our proposed joint iterative detection/decoding scheme with those of other detection methods.

4.3.1 Definitions and Parameters

In this section, we present some simulated BER curves plotted against SNR. The SNR is defined as

$$\text{SNR} = 10 \log_{10} \left(\frac{\sum_{i=0}^{M_h-1} \sum_{j=0}^{N_h-1} (h_{i,j})^2}{2R\sigma^2} \right) \text{ dB} \quad (4.39)$$

where R is the overall code rate and σ^2 is the variance of the AWGN component. In the simulation, we use the following read head response matrices:

$$\mathbf{H}_A = \begin{pmatrix} 0.050684 & 0.21273 & 0.050684 \\ 0.23825 & 1 & 0.23825 \\ 0.050684 & 0.21273 & 0.050684 \end{pmatrix} \quad (4.40)$$

$$\mathbf{H}_B = \begin{pmatrix} 0.0035638 & 0.14843 & 0.0035638 \\ 0.013382 & 0.55733 & 0.013382 \\ 0.0035638 & 0.14843 & 0.0035638 \end{pmatrix} \quad (4.41)$$

$$\mathbf{H}_C = \begin{pmatrix} -0.023 & 0.264 & -0.023 \\ -0.087 & 1 & -0.087 \\ -0.023 & 0.264 & -0.023 \end{pmatrix} \quad (4.42)$$

$$\mathbf{H}_D = \begin{pmatrix} 0.0347 & 0.2297 & 0.0347 \\ 0.1277 & 1 & 0.1277 \\ 0.0347 & 0.2297 & 0.0347 \end{pmatrix} \quad (4.43)$$

where the channel response matrices $\mathbf{H}_A$ and $\mathbf{H}_B$ are the ones used in [92] and $\mathbf{H}_C$ and $\mathbf{H}_D$ are the ones used in [93]. For the performance comparison with the MCMC-based algorithm [92], we use an LDPC code (1998, 1776) with code rate $R = 0.889$, $R_w = 36$ and $C_w = 4$ which has the same parameters as that in [92]. For the performance comparison with the PIE-based algorithm [93], we use an LDPC code (4066, 1776) with code rate $R = 0.895$, $R_w = 38$ and $C_w = 4$ which has the same parameters as that in [93]. These codes are constructed using the semi-random algorithm [19]. A block interleaver is adopted in our simulations.

4.3.2 Effect of Parameters on System Performance

The system performance of the proposed JIDDS scheme will be affected by some parameters such as the size of the data array, the interference level of the 2-D ISI channel, and the numbers of iterations. In the following, we will investigate the effect of these parameters on the system performance.

Firstly, we investigate the effect of the size of the 2-D data array on system performance. In order to simplify the analysis, an uncoded system is used and we simply measure the BER of the deinterleaved LLRs $L_{oc}(c_n), n = 1, \dots, N$ from the cross-track detector. The number of iterations between the down-track detector and cross-track detector is set to $I_{dec} = 3$. The input sequence $\mathbf{u} = \{u_i \in \{-1, 1\}, i = 1, \dots, N\}$ is randomly generated with equal probabilities of -1 s and 1 s. Figure 4.8 shows the simulation results for four combinations of N_r and N_c , namely, $(N_r = 10, N_c = 1000)$, $(N_r = 100, N_c = 100)$, $(N_r = 500, N_c = 20)$, and $(N_r = 20, N_c = 20)$. Since similar results can be observed for channel response matrices $\mathbf{H}_A$ and $\mathbf{H}_B$, we only plotted the simulation results for channel $\mathbf{H}_A$. It can be seen from Fig. 4.8 that these four curves are almost identical. Thus, we can conclude that the size of the 2-D data array has no

Figure 4.8. Effect of the size of the 2-D data array on the system performance.

effect on the system performance.

Next, we examine the effect of the interference level of the 2-D ISI channel on the system performance. We use the uncoded system and a separable channel response matrix given by

$$\mathbf{H} = \begin{pmatrix} ab & b & ab \\ a & 1 & a \\ ab & b & ab \end{pmatrix} \quad (4.44)$$

where a and b represent the interference level in the down-track direction and cross-track direction, respectively. Figure 4.9 shows the BER curves for various values of a and b at SNR=10 dB. Note that here we use SNR=10 dB instead of SNR=16 dB for

Figure 4.9. Effect of the interference level of the 2-D ISI channel on the system performance at SNR=10 dB.

Fig. 3.3 in the previous chapter. This is because in Fig. 3.3, we assume that the binary symbol $x_{i,j}$ is drawn from $\{0, 1\}$. Here, we assume the binary symbol $x_{i,j}$ is drawn from $\{-1, 1\}$. Thus, BER is around 10^{-5} for small interference level at SNR=10 dB as shown in Fig. 4.8. It can be observed from Fig. 4.9 that the BER of the uncoded system does not increase monotonically as a and b increase. The BER becomes the worst when a and b are around 0.6, which is similar to the observation drawn in Fig. 3.3. This is because the system performance is dominated by some small values of $d/2\sigma$. When a and b are around 0.6, there are more small values of $d/2\sigma$. The detailed analysis can be found in the previous chapter.

In Fig. 4.10, we compare the performance of our proposed scheme with different numbers of iterations for the channel response matrix $\mathbf{H}_A$. In this figure, we denote

the numbers of iterations as $I_{dec}/I_{det}/I_{in}/I_{out}$. It can be observed that performing 50 iterations inside the LDPC decoder gives about 0.2 dB performance gain over that of 10 iterations at BER of 10^{-5} . Since there is hardly any performance gain when the number of iterations exceeds 50 inside the LDPC decoder, we set $I_{dec} = 50$ in the subsequent simulations. From Fig. 4.10, it can be observed that larger values of I_{det} , I_{in} and I_{out} can all improve the system performance. We observe that there is hardly any performance improvement when $I_{det} > 3$. The performance gain becomes very small when I_{in} and I_{out} exceed 3. We can also observe that detector iteration is more effective than inner iteration as the scheme of 50/3/1/1 performs about 0.3 dB better than that of 50/1/3/1. The most effective iteration is the outer iteration, as the scheme of 50/1/1/3 has more than 0.2 dB performance gain over that of 50/3/1/1. However, performing outer iteration is also the most complicated one. For each outer iteration, the down-track detector, the cross-track detector, and the LDPC decoder are all implemented once, respectively. When introducing all the iterations, we observe from Fig. 4.10 that the system with parameters of 50/3/3/5 gives the best performance. The consistent results are observed for channel response matrix $\mathbf{H}_B$.

4.3.3 Performance and Complexity Comparisons

Figures 4.11 and 4.12 show the BER performance comparisons of uncoded system for channel response matrices $\mathbf{H}_A$ and $\mathbf{H}_B$, respectively. In these figures, we compare our proposed full-state and reduced-state detectors with the maximum-likelihood page detection (MLPD) bounds and the SSWA. Since in our simulation the channel response matrix has a size of 3×3 , $Q = 3$ corresponds to the full-state detector and $Q = 2$ corresponds to the reduced-state detector. In the SSWA, the signals of adjacent 3 tracks are processed to generate the symbol probabilities based on which the binary bits in

Figure 4.10. Effect of iterations on the system performance for LDPC code (1998,1776).

the middle track are detected [83]. Thus, the performance of SSWA represents the optimal detection performance when only signals of three adjacent tracks are processed to detect the bit sequence in the middle track. Conversely, instead of processing only 3 tracks to detect the middle track, our proposed iterative detector detects the whole page of data cooperatively. It can be observed that our proposed detector has significant performance gain over that of the SSWA. This performance gain is achieved at the cost of higher computational complexity and higher latency. The complexity comparisons among different detection methods are shown in Table 4.1. Note that the computational complexity of our proposed detector should be equal to (complexity of down-track detector + complexity of cross-track detector) $\times I_{det}$ which is higher than that of the SSWA. The computational complexity of the equalizer-based method proposed in [83]

Figure 4.11. BER performance comparisons of uncoded system for channel $\mathbf{H}_A$.

is also shown in Table 4.1. The equalizer-based method adopts a minimum mean-square error (MMSE) equalizer of length $2l$ followed by a modified Viterbi algorithm. The equalizer-based method has linear computational complexity that is proportional to the length of the equalizer but independent of the 2-D ISI channel size. However, the equalizer-based method can only perform equally to or worse than the SSWA [83].

For channel $\mathbf{H}_A$, the BER performance of our proposed detector lies within the lower and upper bounds of MLPD at low SNR but lies slightly higher than the upper bound when SNR becomes large. For channel $\mathbf{H}_B$, the performance of our proposed detector always lies within the lower and upper bounds of MLPD. Thus, we can conclude that the proposed detector performs close to the MLPD. It can also be observed from Figures 4.11 and 4.12 that there are some performance gaps between the reduced-state

Figure 4.12. BER performance comparisons of uncoded system for channel $\mathbf{H}_B$.

detector and full-state detector for channel $\mathbf{H}_A$. However, the reduced-state detector and full-state detector perform almost identically for channel $\mathbf{H}_B$. This is because the reduced-state detector approximates the calculation of $\mathbf{x}_{i,j}^T \cdot \mathbf{h}_0$ by $\hat{\mathbf{x}}_{i,j}^T \cdot \mathbf{h}_0$, where $\hat{\mathbf{x}}_{i,j}$ is determined in the forward recursion. For channel $\mathbf{H}_B$, since $\mathbf{h}_0$ is smaller, the approximation of $\hat{\mathbf{x}}_{i,j}^T \cdot \mathbf{h}_0$ becomes more trivial. Thus, the reduced-state detector performs closer to the full-state detector when the channel coefficients associated with the approximation is smaller.

The BER performance comparisons of different LDPC coded systems are presented in Figures 4.13 and 4.14. In [92], the authors claimed that their proposed MCMC-based detection algorithm outperforms the IRCSDFA which has been reported to have better performance than other suboptimal detection algorithms. Thus, in Fig. 4.13, we com-

Figure 4.13. BER performance comparisons of JIDDS and MCMC-based algorithm.

pare the BER performance and computational complexity of our proposed JIDDS with those of the MCMC-based algorithm. For a fair comparison, both systems adopt the (1998, 1776) LDPC codes, and we set the number of iterations between channel detector and the LDPC decoder to be 5 and the number of iterations inside the LDPC decoder to be 50. Thus, the iteration scheme adopted in our system is 50/3/1/5. It can be observed that for channel $\mathbf{H}_A$, our proposed schemes with full-state and reduced-state detector outperform the MCMC-based method by about 0.4 dB and 0.2 dB, respectively. For channel $\mathbf{H}_B$, the full-state detector and reduced-state detector perform nearly identical, achieving about 0.4 dB performance gain over that of the MCMC-based method. The computational complexity of the MCMC-based algorithm is also shown in Table 4.1, where F_c and N_s are some parameters used in the MCMC-based detection. The typical

Figure 4.14. BER performance comparisons of JIDDS and PIE-based algorithm.

settings are $F_c = 125$ and $N_s = 50$ [92]. It can be observed that the computational complexity of the MCMC-based algorithm grows polynomially with the 2-D interference size while the complexity of our proposed detection algorithm grows exponentially with the 2-D interference size. However, a 3×3 channel response matrix approximates the practical channel model very well and most of the 2-D interference channels investigated in the literature have sizes of 3×3 . For a 3×3 2-D interference channel, the computational complexity of our full-state detection algorithm is comparable to that of the MCMC-based algorithm. The reduced-state detection algorithm even has a lower computational complexity than that of the MCMC-based algorithm. In Fig. 4.13, the performance of (1998, 1776) LDPC code over an AWGN channel is also plotted for reference. The performance of our proposed JIDDS over channel $\mathbf{H}_A$ is about 0.8 dB

away from that of the LDPC code over an AWGN channel. For channel $\mathbf{H}_B$, as the interference level becomes smaller, the performance gap decreases to 0.4 dB. Figure 4.14 shows the performance comparisons of our proposed JIDDS with the PIE-based algorithm [93]. We use the same (4066, 3639) LDPC code for both JIDDS and PIE-based algorithm in this performance comparison. We adopt the 50/3/1/5 iteration scheme in JIDDS. It can be observed from Fig. 4.14 that the JIDDS outperforms the PIE-based algorithm by 2.7 dB and 2.3 dB over channels $\mathbf{H}_C$ and $\mathbf{H}_D$, respectively, at the BER of 10^{-5} . This performance gain is achieved at the cost of higher computational complexity. As the PIE-based algorithm adopts a modified SOVA in the channel detection, the computational complexity of the channel detector in PIE-based algorithm is roughly in the order of 2^{M_h} which is lower than that of the JIDDS. Overall, our proposed JIDDS mitigates the 2-D ISI effectively and performs close to the system with no interference.

In Fig. 4.13, it can be observed that the reduced-state detector performs only 0.2 dB worse than the full-state detector for channel matrix $\mathbf{H}_A$ and almost the same to the full-state detector for channel matrix $\mathbf{H}_B$. From Fig. 4.14, it can also be observed that the reduced-state detector performs 0.05 dB and 0.2 dB worse than the full-state detector over channels $\mathbf{H}_C$ and $\mathbf{H}_D$, respectively, at the BER of 10^{-5} . The reason for these observations has been discussed above. For the 3×3 2-D interference channel considered here, the number of states in the trellis of the down-track detector is 64 and 8 for the full-state detector and reduced-state detector, respectively. As shown in Table 4.1, the overall system complexity is dominated by the down-track detector. Thus, when we adopt the reduced-state down-track detector, the complexity of the channel detector is reduced by roughly 8 times. Considering the large amount of computational complexity it saves and the marginal performance loss, the reduced-state technique is desirable to be adopted in the 2-D iterative detector.

4.4 Conclusion

In this chapter, we have proposed a joint iterative detection/decoding scheme (JIDDS) for a generalized discrete 2-D interference channel. The channel detector is composed of two constituent detectors which work on trellis structure in the down-track direction and cross-track direction, respectively. We have adopted LDPC codes as the error-correcting codes. We have also introduced iterations between the two constituent detectors and among the two constituent detectors with LDPC decoder to achieve better performance. The computational complexity of the system has been analyzed. In order to reduce the system complexity, a reduced-state detector has been proposed. We have also evaluated the effect of array size, interference level, and number of iterations on the system performance. Simulation results have shown that our proposed JIDDS has better BER performance than some other detection/decoding algorithms. The reduced-complexity channel detector greatly reduces the overall system complexity at the expense of only very little performance loss. Our proposed JIDDS has been shown to be very effective in mitigating the 2-D interference while at the same time achieving a satisfactory system performance.

Chapter 5

Joint Message-Passing Decoding Algorithm

Low-density parity-check (LDPC) code is a class of very powerful error-correcting code (ECC). An appealing practical aspect of the decoding of LDPC codes is that it consists of many small, independent decoding functions (i.e., the variable nodes and check nodes). Thus, in practical implementation, the hardware can operate in parallel, leading to a very high-speed decoder. This aspect is very important in magnetic recording systems, where data rates should be high and decoding delays must be low. In Chapter 4, we have proposed a joint iterative detection/decoding scheme (JIDDS) for LDPC coded 2-D ISI channels. However, in JIDDS, three blocks, namely, down-track detector, cross-track detector, and LDPC decoder are implemented sequentially, i.e., we need to wait for the current block to be completed and store its whole output before we can proceed to the next block. Thus, this will result in large memory cost and the LDPC codes advantage of low decoding delay is degraded significantly.

In order to effectively mitigate the 2-D ISI while at the same time make use of the low-delay advantage of LDPC codes, in this chapter, we propose a joint message-passing

decoding algorithm (JMPDA) of LDPC codes and 2-D ISI channels. The JMPDA can be viewed as a windowed version of the JIDDS. In JMPDA, the channel detector consists of many state-processing nodes which can be implemented in a completely parallel manner in conjunction with the parallel decoding of LDPC codes. Thus, the advantages of the JMPDA over JIDDS are higher decoding speed and lower memory cost. Extrinsic information transfer (EXIT) chart is a powerful tool to visualize and analyze the trajectory of iterative decoding system [101]. Different from the density evolution (DE) technique in which an infinite code length must be assumed, the EXIT chart can be used to analyze the iterative decoding system with a finite code length. In this chapter, we introduce EXIT chart to analyze the performance and convergence behavior of the proposed JMPDA. Simulation results are presented to compare the performance of the proposed algorithm with the that of JIDDS and MCMC-based algorithm.

5.1 Joint Decoding Algorithm

We follow the same system model as presented in Chapter 4. Our target is to detect the recorded bits $x_{i,j}$, for $i = 1, 2, \dots, N_r$ and $j = 1, 2, \dots, N_c$, given the received signals $r_{i,j}$, for $i = 1, 2, \dots, N_r$ and $j = 1, 2, \dots, N_c$, that suffer from 2-D ISI and AWGN. The proposed JMPDA consists of two sub-algorithms, namely, the state-processing node update and the variable-check update. In the following, we will explain the detailed algorithm.

5.1.1 State-Procesing Node Update

It has been pointed out in [48] that the key to achieving good performance in the channel detection is to pass state information. In the 2-D ISI channels, the interferences come

Figure 5.1. Message-passing diagram of a state-processing node.

from two orthogonal directions. There is no simple way to incorporate the channel memories to the states in a trellis as that in 1-D ISI channels. Thus, here we define two types of states that take into account the memories from the down-track direction and cross-track direction, respectively. In the down-track direction, a down-track state at the coordinates (i, j) is defined as the previous N_h symbols that interfere with the current symbol, i.e., $S_{i,j} = [\mathbf{x}_{i,j-N_h}, \mathbf{x}_{i,j-N_h+1}, \dots, \mathbf{x}_{i,j-1}]$, where $\mathbf{x}_{i,j} = [x_{i,j}, x_{i-1,j}, \dots, x_{i-M_h+1,j}]^T$. In the cross-track direction, a cross-track state at the coordinates (i, j) is defined as the previous M_h bits that interfere with the current bit, i.e., $s_{i,j} = [x_{i-1,j}, x_{i-2,j}, \dots, x_{i-M_h,j}]$. Thus, the numbers of down-track states and cross-track states are $2^{(M_h+1)N_h}$ and 2^{M_h} , respectively. The down-track state and cross-track state are the same as the states used in the down-track detector and cross-track detector, respectively, in the JIDDS.

The down-track state information and cross-track state information are processed in the state-processing node and transferred in the down-track direction and cross-

track direction, respectively. Figure 5.1 shows the message-passing diagram of a state-processing node at the coordinates (i, j) . The triangle represents a state-processing node which sends and receives four types of messages. Firstly, the state-processing node at the coordinates (i, j) communicates the down-track state information with its two neighboring state-processing nodes at the coordinates $(i, j - 1)$ and $(i, j + 1)$, respectively, in the down-track direction. Secondly, the cross-track state information $s_{i,j}$ is communicated with the two neighboring state-processing nodes at the coordinates $(i - 1, j)$ and $(i + 1, j)$, respectively, in the cross-track direction. The third type of message communicated with the state-processing node is the binary information corresponding to bit $x_{i,j}$ that is feedback from the variable nodes of the LDPC codes. We denote this type of information as $L_{vs}(x_{i,j})$ and $L_{sv}(x_{i,j})$ where the subscripts vs and sv imply this information is transferred from a variable node to a state-processing node or from a state-processing node to a variable node, respectively. Lastly, the state-processing node processes the readback signal $r_{i,j}$ from which we can obtain that

$$\begin{aligned} \log P(r_{i,j}|y_{i,j}) &= -\frac{(r_{i,j} - y_{i,j})^2}{2\sigma^2} \\ &= -\frac{(r_{i,j} - \sum_{n=0}^{N_h} \mathbf{x}_{i,j-n}^T \cdot \mathbf{h}_n)^2}{2\sigma^2}. \end{aligned} \quad (5.1)$$

where $y_{i,j}$ is the output of 2-D ISI channel without AWGN and σ is the standard deviation of the AWGN.

Let the quantities $A_{i,j}^t(p)$ and $B_{i,j}^t(p)$ be the state information that is passed forward from the state-processing node at the coordinates $(i, j - 1)$ to the state-processing node at the coordinates (i, j) and backward from the state-processing node at the coordinates (i, j) to the state-processing node at the coordinates $(i, j - 1)$, respectively, in the down-track direction and the t -th iteration, for $p = 1, \dots, 2^{(M_h+1)N_h}$. The update of $A_{i,j}^t(p)$

and $B_{i,j}^t(p)$ can be implemented as

$$A_{i,j+1}^{t+1}(p) = \max_{p': p' \rightarrow p}^* \{A_{i,j}^t(p') + L_a^t(\mathbf{x}_{i,j}) + \log P(r_{i,j}|y_{i,j})\} \quad (5.2)$$

and

$$B_{i,j}^{t+1}(p') = \max_{p: p' \rightarrow p}^* \{B_{i,j+1}^t(p) + L_a^t(\mathbf{x}_{i,j}) + \log P(r_{i,j}|y_{i,j})\} \quad (5.3)$$

where the function $\max^*$ is defined as in (4.10) and (4.11) in the previous chapter. The notation $p' : p' \rightarrow p$ represents the set of states p' such that $p' \rightarrow p$ is a valid transition. The notation $p : p' \rightarrow p$ is similarly defined. The quantity $L_a^t(\mathbf{x}_{i,j})$ represents the *a priori* log probability of the symbol $\mathbf{x}_{i,j}$ in the t -th iteration. The noiseless component $y_{i,j}$ can be uniquely determined by the pair of states (p', p) . The state information $A_{i,j}^0(p)$ and $B_{i,j}^0(p)$ are initialized such that each state has equal probability at the beginning, i.e.,

$$A_{i,j}^0(p) = B_{i,j}^0(p) = -(M_h + 1)N_h \cdot \log 2, \quad \forall p, i, j. \quad (5.4)$$

The *a posteriori* log probability of the symbol $\mathbf{x}_{i,j}$ in the t -th iteration can be calculated via

$$\log P^t(\mathbf{x}_{i,j}) = \max_{p', p: \mathbf{x}_{i,j}}^* \{A_{i,j}^t(p') + \log P(r_{i,j}|y_{i,j}) + B_{i,j+1}^t(p)\} \quad (5.5)$$

where $p', p : \mathbf{x}_{i,j}$ represents the set of state pairs p' and p corresponding to the symbol $\mathbf{x}_{i,j}$.

Similarly, let the quantities $a_{i,j}^t(q)$ and $b_{i,j}^t(q)$ be the state information in the t -th iteration that is passed upward from the state-processing node at the coordinates

$(i-1, j)$ to the state-processing node at the coordinates (i, j) and downward from the state-processing node at the coordinates (i, j) to the state-processing node at the coordinates $(i-1, j)$, respectively, in the cross-track direction, for $q = 1, \dots, 2^{M_h}$. The update of $a_{i,j}^t(q)$ and $b_{i,j}^t(q)$ can be implemented as

$$a_{i+1,j}^{t+1}(q) = \max_{q':q' \rightarrow q}^* \{a_{i,j}^t(q') + L_a(x_{i,j}) + \log P^t(\mathbf{x}_{i,j})\} \quad (5.6)$$

and

$$b_{i,j}^{t+1}(q') = \max_{q:q' \rightarrow q}^* \{b_{i+1,j}^t(q) + L_a(x_{i,j}) + \log P^t(\mathbf{x}_{i,j})\}. \quad (5.7)$$

Similar to the state information $A_{i,j}^0(p)$ and $B_{i,j}^0(p)$ in the down-track direction, the state information $a_{i,j}^0(q)$ and $b_{i,j}^0(q)$ in the cross-track direction is initialized as

$$a_{i,j}^0(q) = b_{i,j}^0(q) = -M_h \cdot \log 2, \quad \forall q, i, j. \quad (5.8)$$

The *a priori* log probability $L_a^t(\mathbf{x}_{i,j})$ of the symbol $\mathbf{x}_{i,j}$ in the t -th iteration can be updated as

$$L_a^t(\mathbf{x}_{i,j}) = \log(a_{i,j}^t(q') + L_a(x_{i,j}) + b_{i+1,j}^t(q)) \quad (5.9)$$

where the pair of states q' and q and the symbol $\mathbf{x}_{i,j}$ are one-to-one mapping and can be mutually determined.

At the t -th iteration, the state-processing node at the coordinate (i, j) receives the state information $A_{i,j}^t$, $B_{i,j+1}^t$, $a_{i,j}^t$ and $b_{i+1,j}^t$, processes these messages according to the equations (5.2)-(5.9), and transfers the refreshed state information $A_{i,j+1}^{t+1}$, $B_{i,j}^{t+1}$, $a_{i+1,j}^{t+1}$ and $b_{i,j}^{t+1}$ to its neighboring state-processing nodes. This process is implemented for

Figure 5.2. Message-flow diagram of the joint message-passing decoding algorithm.

all the state-processing nodes at all coordinates (i, j) , $i = 1, 2, \dots, N_r$, $j = 1, 2, \dots, N_c$, simultaneously. In the $(t+1)$ -th iteration, the state information $A_{i,j+1}^{t+1}$, $B_{i,j}^{t+1}$, $a_{i+1,j}^{t+1}$ and $b_{i,j}^{t+1}$ will be used as the incoming information to the corresponding state-processing nodes. This is shown in Fig. 5.1.

Note that each state-processing node corresponds uniquely to a recorded bit. Thus, there are N state-processing nodes, each of which is uniquely connected to one variable node of the LDPC code. The message-flow diagram of the joint message-passing decoding algorithm is shown in Fig. 5.2. In this figure, the circles represent the variable nodes and the squares represent the check nodes of the LDPC codes. Each state-processing node is represented by a triangle which is connected to a variable node. The iterations between the state-processing nodes are repeated for T_s times. After that, the state-

processing nodes exchange binary information with the corresponding variable nodes. In the k -th iteration between state-processing nodes and variable node, the binary information passed from the state-processing node to the variable node of LDPC codes is in the form of logarithmic likelihood ratio (LLR) which can be obtained via

$$\begin{aligned}
L_{sv}^k(x_{i,j}) &= \log \frac{P(x_{i,j} = 1)}{P(x_{i,j} = -1)} \\
&= \max_{q', q:1}^* \{a_{i,j}^t(q') + \log P^t(\mathbf{x}_{i,j}) + b_{i+1,j}^t(q)\} \\
&\quad - \max_{q', q:-1}^* \{a_{i,j}^t(q') + \log P^t(\mathbf{x}_{i,j}) + b_{i+1,j}^t(q)\}
\end{aligned} \tag{5.10}$$

where the notation $q', q : 1$ represents the set of state pairs q' and q corresponding to $x_{i,j} = 1$ and $q', q : -1$ is similarly defined.

Equations (5.1)-(5.10) come from the channel detection algorithm of the JIDDS. The difference here is that the signal processing for all the recorded bits in the 2-D data array can be implemented in parallel. Thus, the LLR values $L_{sv}^k(x_{i,j})$ of all bits $x_{i,j}$ are generated simultaneously.

5.1.2 Variable-Check Node Update

As shown in Fig. 5.2, the other parts of the decoding system are the variable and check nodes. The target of the variable-check node update is to execute the LDPC decoding algorithm in parallel and produce more reliable soft estimation of each bit $x_{i,j}$, given the LLR information $L_{sv}^k(x_{i,j})$ from each state-processing node. The variable-check node update algorithm consists of many independent variable node updates and check node updates. In the k -th iteration between state-processing nodes and variable node and the l -th iteration inside the LDPC decoder, the LLR $z_{i,j,m}^{l,k}$ of the bit $x_{i,j}$ which is sent

from the variable node $x_{i,j}$ to check node m can be calculated as

$$z_{i,j,m}^{l,k} = L_{sv}^k(x_{i,j}) + \sum_{n \in D_v(i,j) \setminus m} q_{n,i,j}^{l-1,k} \quad (5.11)$$

where $D_v(i,j) \setminus m$ represents all the check nodes connected to the variable node $x_{i,j}$ except for the check node m . The above variable node updates are executed for all the variable nodes $x_{i,j}$, for $i = 1, 2, \dots, N_r$ and $j = 1, 2, \dots, N_c$, simultaneously. The variable $q_{m,i,j}^{l,k}$ is the LLR of bit $x_{i,j}$ which is sent from the check node m to variable node $x_{i,j}$. The update of $q_{m,i,j}^{l,k}$ can be calculated as

$$q_{m,i,j}^{l,k} = 2 \tanh^{-1} \left\{ \prod_{(i',j') \in D_c(m) \setminus (i,j)} \tanh[z_{i',j',m}^{l,k}/2] \right\} \quad (5.12)$$

where $D_c(m) \setminus (i,j)$ represents all the variable nodes connected to the check node m except for the variable node $x_{i,j}$. The above check node updates are executed for all the check nodes m , for $m = 1, 2, \dots, M$ with $M = N - K$, simultaneously. After T_c times execution of the variable node updates and check node updates, the refreshed LLR value $L_{vs}^k(x_{i,j})$ is transferred from the variable node $x_{i,j}$ to the state-processing node at the coordinates (i,j) which can be obtained as

$$L_{vs}^k(x_{i,j}) = \sum_{n \in D_v(i,j)} q_{n,i,j}^{T_c,k} \quad (5.13)$$

where $D_v(i,j)$ represents all the check nodes connected to the variable node $x_{i,j}$. The LLR values $L_{vs}^k(x_{i,j})$ for all the bits $x_{i,j}$, for $i = 1, 2, \dots, N_r$ and $j = 1, 2, \dots, N_c$, are generated simultaneously and used to further update the *a priori* information $L_a^t(\mathbf{x}_{i,j})$

and $L_a(x_{i,j})$. The quantities $L_a^t(\mathbf{x}_{i,j})$ and $L_a(x_{i,j})$ have a relationship that

$$L_a^t(\mathbf{x}_{i,j}) = \sum_{n=0}^{M_h} L_a(x_{i-n,j}). \quad (5.14)$$

The state-processing nodes and the variable nodes exchange the information for T_o rounds. Finally, the LLR value of the bit $x_{i,j}$ is obtained via

$$L^{T_o}(x_{i,j}) = L_{sv}^{T_o}(x_{i,j}) + \sum_{n \in D_v(i,j)} q_{n,i,j}^{T_c, T_o}. \quad (5.15)$$

5.1.3 Summary of the JMPDA

The joint message-passing decoding algorithm can be summarized as:

- Initialization:
 1. Calculate $\log P(r_{i,j}|y_{i,j})$ according to (5.1), $\forall i, j$;
 2. Set all the initial message $A_{i,j}^0(p)$ and $B_{i,j}^0(p)$ as (5.4), $a_{i,j}^0(q)$ and $b_{i,j}^0(q)$ as (5.8), and $L_a^0(\mathbf{x}_{i,j}) = L_a(x_{i,j}) = 0$, $\forall i, j$;
 3. Set $k = 0$.
- Joint message-passing decoding:
 1. Set $t = 0$;
 2. For all state-processing nodes, process their incoming state information $A_{i,j}^t$, $B_{i,j+1}^t$, $a_{i,j}^t$ and $b_{i+1,j}^t$ as (5.2)-(5.9) simultaneously and transfer the output state information $A_{i,j+1}^{t+1}$, $B_{i,j}^{t+1}$, $a_{i+1,j}^{t+1}$ and $b_{i,j}^{t+1}$ to their neighboring state-processing nodes;

3. Repeat step 2 for T_s times, let $t = t + 1$ for each iteration;
4. Obtain the LLR values $L_{sv}^k(x_{i,j})$ as (5.10), $\forall i, j$, and transfer this information from each state-processing node to the corresponding variable node;
5. Set $l = 0$;
6. For all the variable nodes and check nodes, execute the variable node updates using (5.11) and check node updates using (5.12);
7. Repeat step 6 for T_c times, let $l = l + 1$ for each iteration;
8. Obtain the LLR values $L_{vs}^k(x_{i,j})$ as (5.13), $\forall i, j$, and transfer this information from each variable node to the corresponding state-processing node;
9. Use $L_{vs}^k(x_{i,j})$ to update the *a priori* information $L_a^t(\mathbf{x}_{i,j})$ and $L_a(x_{i,j})$ in the state processing nodes;
10. Repeat steps 1 to 9 for T_o times, let $k = k + 1$ for each iteration.

- Decode

1. Decode the recorded bits $x_{i,j}$, $\forall i, j$, according to $L^{T_o}(x_{i,j})$ which is obtained using (5.15).

Note that for a fix number of iteration T_s in the state-processing node update, a bit is detected based on the adjacent bits which form a window of size $(2T_s + 1) \times (2T_s + 1)$. When T_s is increased, more information is used to detect one bit, thus the reliability becomes higher but the corresponding delay and computational cost also increase.

5.2 EXIT Chart of the Iterative Decoding System

In this section, we introduce the EXIT chart which is a useful tool to trace the asymptotic convergence behavior of iterative decoding systems [101]. The proposed iterative decoding system consists of two constituent parts, namely, the state-processing nodes and the variable-check nodes of the LDPC codes. The relationships of the input and output mutual information of each constituent part are produced independently. Thus, an EXIT chart consists of two curves based on which a trajectory of the iterative decoding system can be drawn and the asymptotic convergence behavior of the system can be observed.

5.2.1 Validation of Gaussian Approximation

In order to facilitate the EXIT chart analysis, we will first validate the Gaussian approximation. Let the LLR values passed from state-processing nodes to variable nodes be denoted as L_{sv}^k and let the LLR values passed from variable nodes to state-processing nodes be denoted as L_{vs}^k in the k -th iteration between state-processing nodes and the variable nodes. During the iterations, the LLR values L_{vs}^k and L_{sv}^k should maintain a Gaussian-like distribution [101] with

$$LLR = \mu \cdot x + n \quad (5.16)$$

where x is the recorded bit, n is a Gaussian distributed variable with zero mean and variance 2μ .

In order to verify this assumption, we obtain some numerical results. We adopt the LDPC code (1008, 504) [102] and 2-D ISI channels $\mathbf{H}_A$ and $\mathbf{H}_B$ which are defined in (4.40) and (4.41), respectively. We set SNR= 3 dB, $T_s = 3$ and $T_c = 3$. The values

Figure 5.3. Probability density function $f(\phi)$ of the normalized values of $L_{vs}^3 \cdot x$ and $L_{sv}^3 \cdot x$.

are averaged over a large number of samples. Figures 5.3 and 5.4 show the probability density function (pdf) $f(\phi)$ and cumulative density function (cdf) $F(\phi)$, respectively, of the normalized values of $L_{vs}^3 \cdot x$ and $L_{sv}^3 \cdot x$. The normalized values of $L_{vs}^3 \cdot x$ are obtained via

$$\phi = \frac{L_{sv}^3 \cdot x - E(L_{sv}^3 \cdot x)}{\sqrt{E((L_{sv}^3 \cdot x - E(L_{sv}^3 \cdot x))^2)}} \quad (5.17)$$

where $E(\cdot)$ represents the expectation operation. The components $E(L_{sv}^k \cdot x)$ and $\sqrt{E((L_{sv}^k \cdot x - E(L_{sv}^k \cdot x))^2)}$ are the mean and standard deviation of $L_{sv}^k \cdot x$, respectively. The normalized values of $L_{vs}^k \cdot x$ are obtained in the same way.

In Figs. 5.3 and 5.4, we also include the pdf and cdf of normal distribution which is a Gaussian function with zero mean and variance of 1. It can be observed that the

Figure 5.4. Cumulative density function $F(\phi)$ of the normalized values of $L_{vs}^3 \cdot x$ and $L_{sv}^3 \cdot x$.

pdf and cdf of the normalized values of $L_{vs}^k \cdot x$ and $L_{sv}^k \cdot x$ are very close to those of normal distribution. Thus, this result verifies that L_{vs}^k and L_{sv}^k well approximate the Gaussian-like distribution.

Table 5.1 shows the average means and variances of $L_{vs}^k \cdot x$ and $L_{sv}^k \cdot x$, for $k = 1, \dots, 5$. It can be observed that the mean values approximately equal to half of the variance values. These results validate the assumption that the LLR values L_{vs}^k and L_{sv}^k satisfy the Gaussian approximation as shown in (5.16).

Table 5.1. LLR values during the k -th iteration

	channel $\mathbf{H}_A$				channel $\mathbf{H}_B$			
	$L_{sv}^k \cdot x$		$L_{vs}^k \cdot x$		$L_{sv}^k \cdot x$		$L_{vs}^k \cdot x$	
k	mean	variance	mean	variance	mean	variance	mean	variance
1	2.85	5.96	2.01	4.44	3.25	6.62	3.22	6.94
2	3.32	6.72	3.74	7.94	3.68	7.52	5.29	11.11
3	3.55	7.14	4.87	9.98	3.81	7.73	6.07	12.45
4	3.66	7.34	5.56	11.04	3.84	7.79	6.28	12.80
5	3.70	7.42	5.85	11.51	3.85	7.80	6.34	12.89

5.2.2 Computation of the Mutual Information

For an array of binary bits whose LLR values approximate Gaussian-like distribution as (5.16), the pdf of the LLR values satisfies

$$p(\xi|x) = \frac{e^{-((\xi-\mu \cdot x)^2/4\mu)}}{\sqrt{4\pi \cdot \mu}} \quad (5.18)$$

which has the condition that

$$p(\xi|x) = p(-\xi|x) \cdot e^{x \cdot \xi}. \quad (5.19)$$

For each constituent part, in order to measure the amount of information between the recorded data and their input LLR values during the iterative decoding, we define the mutual information as

$$I_i = \frac{1}{2} \cdot \sum_{x=-1,1} \int_{-\infty}^{\infty} p(\xi|x) \cdot \log_2 \frac{2p(\xi|x)}{p(\xi|x=-1) + p(\xi|x=1)} d\xi. \quad (5.20)$$

With (5.18) and (5.19), (5.20) becomes

$$I_i(\mu) = 1 - \int_{-\infty}^{\infty} \frac{e^{-((\xi-\mu)^2/4\mu)}}{\sqrt{4\pi \cdot \mu}} \cdot \log_2(1 + e^{-\xi}) d\xi. \quad (5.21)$$

Note that $I_i(\mu)$ is monotonically increasing with μ and thus reversible. The mutual information has a property that

$$\lim_{\mu \rightarrow 0} I_i(\mu) = 0, \quad \lim_{\mu \rightarrow +\infty} I_i(\mu) = 1, \quad \mu > 0. \quad (5.22)$$

We can also define the mutual information between the recorded data and output LLR values of each constituent part, which can be expressed as

$$I_o = \frac{1}{2} \cdot \sum_{x=-1,1} \int_{-\infty}^{\infty} f(\xi|x) \cdot \log_2 \frac{2f(\xi|x)}{f(\xi|x=-1) + f(\xi|x=1)} d\xi \quad (5.23)$$

where $f(\xi|x)$ is the pdf of the output LLR value which is obtained by Monte Carlo simulation.

The state-processing node updates are dependent on the readback signals $r_{i,j}$, for $i = 1, \dots, N_r$ and $j = 1, \dots, N_c$, and the feedback LLR values L_{vs}^k . For a given channel $\mathbf{H}$, mutual information I_i , and SNR value, we can firstly randomly generate $x_{i,j}$ and then obtain the readback signals $r_{i,j}$ and the feedback LLR values L_{vs}^k . Next, we can use Monte Carlo simulation to get the output LLR values L_{sv}^k from which we can obtain the pdf $f(\xi|x)$. The mutual information I_o for variable-check nodes can be obtained similarly. The only difference is that the variable-check node updates in the LDPC code are only dependent on the input LLR values L_{sv}^k from the state-processing nodes but not on the readback signal. Thus, we only need to randomly generate L_{sv}^k and use Monte Carlo simulation to get L_{vs}^k from which we can obtain the corresponding pdf.

5.2.3 EXIT Chart Generating Algorithm

The EXIT chart can be used to visualize the exchange of information during the iterative decoding. In the EXIT chart, there are two curves representing the relationships of the mutual information between the input and output for the state-processing nodes and variable-check nodes, respectively. The algorithm of generating the EXIT chart is summarized as follows:

- For the state-processing nodes:
 1. Randomly choose an array $\mathbf{x} = \{x_{i,j}, i = 1, \dots, N_r, j = 1, \dots, N_c\}$. For a given SNR value, generate an array of AWGN $\mathbf{v} = \{v_{i,j}, i = 1, \dots, N_r, j = 1, \dots, N_c\}$. For a fixed channel response matrix, the readback array $\mathbf{r} = \{r_{i,j}, i = 1, \dots, N_r, j = 1, \dots, N_c\}$ can be obtained from $\mathbf{x}$ and $\mathbf{v}$ via (4.2);
 2. For a given input mutual information $I_i(\mu)$, compute the mean value μ . Randomly generate an array of Gaussian variables with zero mean and variance 2μ . The input LLR values L_{vs}^k can be obtained via (5.16);
 3. Based on readback signal $\mathbf{r}$ and input LLR values L_{vs}^k , execute the state-processing update algorithm (5.1)-(5.10) for T_s times and collect the generated values L_{sv}^k ;
 4. Repeat steps 1 to 3 for sufficient number of times. Obtain the pdf $f(\xi|x)$ of the LLR values L_{sv}^k and calculate the output mutual information I_o according to (5.23);
 5. For different input mutual information $I_i \in [0, 1]$, repeat steps 1 to 4 to obtain the corresponding output mutual information I_o . The mutual infor-

mation curve for state-processing nodes is plotted by putting input I_i on the abscissa and output I_o on the ordinate.

- For the variable-check nodes:

1. For a given input mutual information $I_i(\mu)$, compute the mean value μ . Randomly generate an array $\mathbf{x} = \{x_{i,j}, i = 1, \dots, N_r, j = 1, \dots, N_c\}$ and Gaussian variables with zero mean and variance 2μ . The input LLR values L_{sv}^k can be obtained via (5.16);
2. Based on LLR values L_{sv}^k , execute the variable-check node update algorithm (5.11)-(5.13) for T_c times and collect the generated values L_{vs}^k ;
3. Repeat steps 1 to 2 for sufficient number of times. Obtain the pdf $f(\xi|x)$ of the LLR values L_{vs}^k and calculate the output mutual information I_o according to (5.23);
4. For different input mutual information $I_i \in [0, 1]$, repeat steps 1 to 3 to obtain the corresponding output mutual information I_o . The mutual information curve for state-processing nodes is plotted by putting input I_o on the abscissa and output I_i on the ordinate.

Initially, the state-processing nodes work with zero *a priori* information ($I_i = 0$) and produce L_{sv}^0 purely based on the readback signal. The output L_{sv}^0 of state-processing nodes is passed to the variable nodes and used in the variable-check node updates. The generated output L_{vs}^0 of variable-check nodes is further fed back to the state-processing nodes and serve as *a priori* information in the next iteration. In other words, the output of state-processing nodes serves as the input to the variable-check nodes and the output of the variable-check nodes serves as the input to the state-processing nodes.

Figure 5.5. EXIT chart and trajectory of the joint message-passing decoding system.

Thus, in the EXIT chart, the abscissa and ordinate for the state-processing nodes and variable-check nodes are reversed. The iterative process between the state-processing nodes and variable-check nodes is repeated until the number of iterations exceeds T_o or no mutual information is further updated, which corresponds to an intersection of the two EXIT curves in the EXIT chart.

Using the previously introduced algorithm, an example of EXIT chart is generated and plotted in Fig. 5.5. For the EXIT curve of the state-processing nodes, we set SNR=3.5 dB, $T_s = 3$ and adopt the channel $\mathbf{H}_A$. The EXIT curve of the variable-check nodes is based on the LDPC code (1008, 504) with $T_c = 8$. A trajectory of the iterative decoding system is also plotted in Fig. 5.5. The trajectory is obtained by free-running the simulation and processing the LLR values L_{sv}^k and L_{vs}^k . In plotting the trajectory, we

do not assume Gaussian-like distribution. We obtain the pdfs based on the simulation values of L_{sv}^k and L_{vs}^k and track the mutual information during each iteration according to (5.20). It can be observed in Fig. 5.5 that the trajectory matches well with the EXIT curves. This confirms the validity of EXIT chart in analyzing the behavior of the proposed iterative decoding system. Note that the EXIT curves of state-processing nodes and variable-check nodes are generated independently. Thus, in the following, we can investigate the convergence behavior of the proposed iterative decoding system by analyzing the individual EXIT curve independently.

5.3 Results and Discussions

In this section, we use the EXIT chart to analyze the convergence behavior of the joint message-passing decoding system. Simulation results are also presented for the optimization of the iteration scheme and performance comparisons.

5.3.1 Effects of Parameters on the EXIT Curves

In the EXIT chart, a larger value of the mutual information I_i means the bits become known with higher confidence. Thus, the iterative decoding system will produce correct decisions of the recorded bits if and only if the two EXIT curves intersect at $I_i = 1$ but nowhere else. This requires the EXIT curve of the state-processing nodes lies higher and the EXIT curve of the variable-check nodes lies flatter and lower in the EXIT chart. In the following, we analyze the effects of parameters on the individual EXIT curves which determine the convergence behavior and performance of the iterative decoding system.

There are three parameters that can affect the EXIT curve of the state-processing nodes, namely, the number of iterations T_s , the SNR and the channel response matrix.

Figure 5.6. Effects of T_s on the EXIT curve of state-processing nodes.

The effects of these three parameters are demonstrated in Figs. 5.6-5.8, respectively. Firstly, in Fig. 5.6 we adopt channel $\mathbf{H}_A$ and fix SNR= 3 dB to analyze the effect of T_s . It can be observed that the EXIT curves for different T_s are generally in parallel. When T_s is increased, the EXIT curve moves up. But when $T_s \geq 3$, increasing T_s can hardly change the EXIT curve. Some EXIT curves with channel $\mathbf{H}_A$ and $T_s = 3$ but different values of SNR are plotted in Fig. 5.7. It can be observed that the EXIT curves for different SNR levels are in parallel and equal differences in SNR result in almost equal intervals in the EXIT curves. Lastly, we fix SNR= 4 dB and $T_s = 3$ and compare the EXIT curve of channel $\mathbf{H}_A$ and that of channel $\mathbf{H}_B$. It can be observed that as the interference level of $\mathbf{H}_B$ is smaller, the corresponding EXIT curve lies higher.

There are two parameters that can affect the EXIT curve of the variable-check

Figure 5.7. Effects of SNR on the EXIT curve of state-processing nodes.

nodes of the LDPC codes. The first one is the code structure, including code rate, code length, column-weight, row-weight, etc., which are predetermined at the encoding part and cannot be controlled at the decoding side. The other parameter is the number of iterations T_c which can be controlled at the decoding part. Figure 5.9 shows the effects of the two parameters on the EXIT curve of the variable-check nodes. Here we adopt two LDPC codes, namely (1008, 504) and (1998, 1776), with code rate of 0.5 and 0.89, respectively [102]. It can be observed that as the LDPC code (1998, 1776) has a higher code rate, it lies higher in the EXIT chart. It has been proved in [103] that the area under the EXIT curve of an ECC roughly equals to the code rate of the ECC. Thus, if we combine an LDPC code of higher code rate with the state-processing nodes, a higher value of SNR is required in order to make the iterative decoding system converge to

Figure 5.8. Effects of channel response matrix on the EXIT curve of state-processing nodes.

the correct result. It can also be observed that a larger number of iterations T_c in the decoding of LDPC code can make the EXIT curve flatter. But the difference becomes smaller as T_c increases. Moreover, the number of iteration T_c has a bigger impact on the EXIT curve for LDPC code (1008, 504) than that for LDPC code (1998, 1776).

5.3.2 Performance Comparisons

There are three types of iterations introduced in the proposed JMPDA, namely, the iterations between the state-processing nodes T_s , the iterations between the variable and check nodes T_c , and the outer iterations between the state-processing nodes and variable-check nodes T_o . As shown in the previous results, the system performance can generally improve as the number of iterations increases. We are also interested in the

Figure 5.9. Effects of parameters on the EXIT curve of variable-check nodes.

relationship between the numbers of iterations T_s , T_c and T_o that can be combined to produce the best performance. Since T_o is the number of outer iterations in which the state-processing iterations and variable-check iterations are executed for T_s and T_c times, respectively, we fix $T_o(T_s + T_c)$ to 50 and optimize the number of iterations by simulation. Figure 5.10 shows some BER results for different combinations of T_s , T_c and T_o and we denote the numbers of iterations as $T_c/T_s/T_o$ in the figure. It can be observed that for the LDPC code (1008, 504), the best performance is achieved by setting $T_c = 8$, $T_s = 2$ and $T_o = 5$. The combination of 7/3/5 performs slightly worse than 8/2/5 and the combination of 3/2/10 performs more than 1 dB worse than the other two iteration schemes. For the LDPC code (1998, 1776), the three different iteration schemes perform very closely. Among the three iteration schemes, 7/3/5 has the best BER performance

Figure 5.10. Simulation results for different combinations of number of iterations.

but the performance gain is very marginal comparing with the other two. The reason why the LDPC codes (1008, 504) and (1998, 1776) perform so differently is because the decoding of LDPC code (1008, 504) is very sensitive to the number of iterations T_c as shown in Fig. 5.9. Thus different values of T_c can result in big performance differences for LDPC code (1008, 504). On the other hand, the number of iterations T_c does not affect much on the decoding of LDPC code (1998, 1776) as shown in Fig. 5.9 and thus different iteration schemes perform closely for the LDPC code (1998, 1776).

Next we compare the BER performance of the proposed JMPDA with that of the JIDDS and MCMC-based algorithm. For a fair comparison, in all systems, the LDPC code (1998, 1776) is adopted with 50 decoding iterations and the number of outer iterations between channel detector and LDPC decoder is set to 5. Figure 5.11 shows

Figure 5.11. Performance comparisons of the proposed JMPDA with that of the JIDDS and MCMC-based algorithm.

the BER performance comparisons of the these systems based on both channels $\mathbf{H}_A$ and $\mathbf{H}_B$. It can be observed that as the interference level is smaller for channel $\mathbf{H}_B$, all systems perform better with channel $\mathbf{H}_B$ than with channel $\mathbf{H}_A$. For both channels $\mathbf{H}_A$ and $\mathbf{H}_B$, the proposed JMPDA performs nearly identical to the JIDDS and achieves about 0.3 dB performance gain over the MCMC-based algorithm at BER of 10^{-5} . This demonstrates the superior performance of our proposed JMPDA over 2-D ISI channels.

5.4 Conclusion

In this chapter, we have proposed a JMPDA to decode LDPC codes over 2-D ISI channels. The iterative decoding algorithm mainly consists of two types of nodes,

namely, state-processing nodes and variable-check nodes. In the state-processing node update, two types of state information are passed in the down-track direction and cross-track direction, respectively. All state-processing nodes generate the soft information of the corresponding bits and pass them to the variable nodes simultaneously. A parallel LDPC decoding algorithm is executed which generates extrinsic information and feeds back the information to the state-processing nodes. An advantage of the JMPDA is that the algorithm can be implemented in a fully parallel manner, leading to a high-speed decoder. The EXIT chart analysis is also introduced in this chapter to analyze the convergence behavior of the proposed decoding algorithm. In the EXIT chart, two EXIT curves corresponding to the state-processing nodes and variable-check nodes are obtained independently. The effects of parameters to the decoding system have been investigated by analyzing the individual EXIT curves. Finally, simulation results have been presented for the optimization of the iteration scheme and performance comparisons. It has been shown that the proposed JMPDA performs nearly identical to the JIDDS and outperforms the MCMC-based algorithm.

Chapter 6

Performance Analysis of LDPC Codes over 2-D ISI Channels via Density Evolution

It has been shown in [23] that LDPC codes exhibit a waterfall phenomenon over memoryless channels, i.e., there exists a noise threshold value below which the BER performance will suddenly drop to zero. Density evolution (DE) technique has been proposed in [23] to optimize the degree distribution of LDPC codes and hence increase the threshold values. In [62], the authors proposed a modified DE algorithm to calculate the threshold values of LDPC codes over binary linear inter-symbol interference (ISI) channels.

For 2-D ISI channels, we are interested in two questions: 1). Do LDPC codes also exhibit a waterfall phenomenon over 2-D ISI channels? 2). What is the best performance that the LDPC coded 2-D ISI system can achieve? These two questions have not been answered in the literature, and they are addressed in this chapter. Although the DE algorithm has been adopted to calculate the noise thresholds in [90], the au-

thors did not prove or even agree to the concentration results. They claimed that the concentration results do not hold due to the short cycles in the channel graph of their low-complexity joint equalization and decoding scheme (LCJEDS). At the same time, the authors showed in [90] that the simulation results for LCJEDS respect the thresholds calculated through the DE algorithm. Since it has been shown in the previous chapter that the joint message-passing decoding algorithm (JMPDA) is a windowed version of the joint iterative detection/decoding scheme (JIDDS) and performs nearly identical to the JIDDS which outperforms other detection/decoding algorithms, in this chapter, we adopt the JMPDA in the detection of LDPC coded 2-D ISI systems. Via the graph representation of the detection algorithm, we will prove a concentration result that as the code length tends to infinity, the message-flow neighborhood will become tree-like and the average performance of the system is closely concentrated around the expected performance which exhibits a waterfall phenomenon. By using a modified DE algorithm, a threshold is then calculated and can be considered as the supremum of performance for error-free transmission. When the codeword length is long enough, error-free performance can be achieved if and only if the parameter of the channel is below the calculated threshold. Simulation results verify that the thresholds can accurately predict the performance of the JMPDA for large block lengths.

6.1 System Model

The read channel model of an LDPC coded 2-D ISI channel is similar to that in Chapter 4. Here we consider a coset code of LDPC codes which is specified by a parity-check matrix $\mathbf{H}_c$ and a $1 \times N$ coset-defining vector $\mathbf{b}$. We can obtain the generator matrix

$\mathbf{G}_c$ from $\mathbf{H}_c$ as (2.8) and (2.9). The output of a typical LDPC encoder is given by

$$\mathbf{s} = [s_1, s_2, \dots, s_N] = (\mathbf{u} \cdot \mathbf{G}_c) \oplus \mathbf{b} \quad (6.1)$$

where $\oplus$ represents binary addition. The codeword $\mathbf{s}$ satisfies the condition of

$$\mathbf{s} \cdot \mathbf{H}_c^T = \mathbf{d} = [d_1, d_2, \dots, d_M] \quad (6.2)$$

where $M = N - K$ is the length of parity-check bits of the code. The superscript T represents the transpose operation. The code is linear if and only if $\mathbf{d} = \mathbf{0}$; otherwise, this code is a coset code of an LDPC code. The node degree distribution of the code is specified by two degree polynomials given by [22]

$$\lambda(x) = \sum_{i=1}^{d_v^{max}} \lambda_i x^{i-1} \quad (6.3)$$

and

$$\rho(x) = \sum_{i=1}^{d_c^{max}} \rho_i x^{i-1} \quad (6.4)$$

where d_v^{max} and d_c^{max} are the maximal variable-node and check-node degrees, respectively. The parameters λ_i and ρ_i represent the fraction of edges with variable-node degree i and check-node degree i , respectively. Clearly, we have

$$\sum_{i=1}^{d_v^{max}} \lambda_i = \sum_{i=1}^{d_c^{max}} \rho_i = 1. \quad (6.5)$$

The code rate R of the LDPC code is

$$R = \frac{K}{N} = 1 - \frac{\int_0^1 \rho(x) dx}{\int_0^1 \lambda(x) dx}. \quad (6.6)$$

A regular LDPC code is a code for which $\lambda_{d_v^{max}} = \rho_{d_c^{max}} = 1$. We denote N_e as the total number of variable-to-check edges in the graph of LDPC codes.

We adopt the JMPDA to decode the LDPC coded 2-D ISI system. Since we consider a coset code of LDPC code, the check node update in (5.12) is modified as

$$q_{m,i,j}^{l,k} = 2 \tanh^{-1} \left\{ (-1)^{d_m} \prod_{(i',j') \in D_c(m) \setminus (i,j)} \tanh[z_{i',j',m}^{l,k}/2] \right\}. \quad (6.7)$$

The other parts of the decoding system remain the same as those in the JMPDA.

6.2 Concentration, Density Evolution and Threshold Calculation

In this section, we will first explain the message-flow neighborhood structure of the iterative detection/decoding system. We will then prove a concentration statement for every possible input sequence, and we will show that the average performance of the system is closely concentrated around the system performance when the input sequence is independent and uniformly distributed (i.u.d.). Based on the concentration result, we will introduce the density evolution method to track the message densities in the iterations. Finally, using the density evolution technique, we can calculate a threshold which can be considered as the performance limit of the system.

Figure 6.1. Message-flow of $L_{sv}^0(x_{i,j})$ and its equivalent representation with $T_s = 1$.

6.2.1 Message-Flow Neighborhoods

In the state-processing node update of the JMPDA, the state information $A_{i,j}^0(p)$, $B_{i,j}^0(p)$, $a_{i,j}^0(q)$, and $b_{i,j}^0(q)$, $\forall i, j$, is initialized such that each state has an equal probability. During the iterations between the state-processing nodes, each state-processing node only exchanges state information with its adjacent state-processing nodes. Thus, for a fixed number of iterations between state-processing nodes denoted as T_s , the output soft decision of a certain bit $x_{i,j}$ is only dependent on the information of its neighboring bits which form a window of size $(2T_s + 1) \times (2T_s + 1)$, i.e., the LLR value $L_{sv}^0(x_{i,j})$ is calculated based on the information corresponding to the bits $\{x_{i',j'} : i' = i - T_s, \dots, i + T_s, j' = j - T_s, \dots, j + T_s\}$ with $T_s < \infty$. Each bit in this window also corresponds to a variable node. Figure 6.1 illustrate the message-flow of

Figure 6.2. Message-flow neighborhood of a variable node to a check node and its equivalent representation with $k = 1$ and $T_c = 1$.

$L_{sv}^0(x_{i,j})$ with $T_s = 1$. In this figure, the triangles and the circles represent the state-processing nodes and variable nodes, respectively. The equivalent representation of the message-flow is also shown in Fig. 6.1 in which we use a dark triangle to represent the state-processing node update.

For clarity of presentation, we consider regular LDPC codes, where each variable node has degree $d_v = d_v^{max}$ and each check node has degree $d_c = d_c^{max}$. Consider an edge $\vec{e}$ that connects the variable node V_e to the check node C_e at the end of the k -th iteration between the state-processing nodes and the variable-check nodes. The message-flow neighborhood of depth k of the edge $\vec{e}$, denoted by $N_{\vec{e}}^k$, is a subgraph that consists of the two nodes V_e and C_e , the edge $\vec{e}$, and all nodes and edges that contribute

to the computation of the message $z_{V_e, C_e}^{T_c, k}$ passed from V_e to C_e . Denote x_{V_e} as the bipolar bit corresponds to the variable node V_e . Recall that $z_{V_e, C_e}^{T_c, k}$ is computed as

$$z_{V_e, C_e}^{T_c, k} = L_{sv}^k(x_{V_e}) + \sum_{n \in D_v(V_e) \setminus C_e} q_{n, V_e}^{T_c-1, k} \quad (6.8)$$

which consists of two parts, namely, $L_{sv}^k(x_{V_e})$ and $\sum_{n \in D_v(V_e) \setminus C_e} q_{n, V_e}^{T_c-1, k}$. The first part comes from the state-processing node corresponding to V_e . As discussed above, the computation of $L_{sv}^k(x_{V_e})$ is dependent on the $4T_s(T_s + 1)$ neighboring bits of x_{V_e} . Each of these neighboring bits also corresponds to a variable node which is further constrained by T_c stages of check nodes and variable nodes. The second part comes from $d_v - 1$ check nodes which are also connected to $T_c - 1$ stages of variable nodes and check nodes. Figure 6.2 shows a depth-1 message-flow neighborhood and the equivalent representation with $k = 1$ and $T_c = 1$. In this figure, the squares represent the check nodes. The message-flow neighborhood of depth k can be obtained by branching out the last-stage variable nodes of the depth-1 neighborhood as shown in Fig. 6.3.

Proposition 6.1: When the code graph is randomly chosen with uniform distribution from the ensemble of all graphs, then for a sufficiently large value of N , the message-flow neighborhood $\mathbb{N}_e^{k*}$ will form a tree with the probability that

$$\Pr\{\mathbb{N}_e^{k*} \text{ is not tree-like}\} \leq \eta/N \quad (6.9)$$

for some constant η which is independent of N . The number k^* is a fixed depth which represents the number of iterations between the state-processing nodes and the variable-check nodes.

Proof: Please refer to Appendix A. ■

Figure 6.3. Message-flow neighborhood of depth k .

6.2.2 Concentration Theorems

As shown in Proposition 6.1, for a sufficiently large value of N , the message-flow neighborhood will be tree-like. The tree can be specified by the binary bits of variable nodes in the tree. We denote $N_v(k)$ as the number of possible fillings of binary bits in the tree-like neighborhood of depth k . Each possible filling is referred to as a message-flow

neighborhood type. We index these neighborhoods as

$$\theta_i \in \{0, 1\}^{N_v(k)}, \text{ where } 1 \leq i \leq N_v(k). \quad (6.10)$$

Let c_θ represent the original bipolar bit corresponding to the variable node V_e which is at the top of the message-flow neighborhood of type θ . Let $p^k(\theta)$ be the probability that a tree of type θ and depth k delivers an incorrect c_θ , i.e.,

$$p^k(\theta) = \Pr(z_{V_e, C_e}^{T_e, k} \cdot c_\theta < 0 | \text{tree type } \theta). \quad (6.11)$$

We denote $\mathbf{c}$ as a particular sequence of transmitted bits. For a given edge $\vec{e}$ whose message-flow neighborhood of depth k is tree-like, let $p^k(\mathbf{c})$ be the expected number of incorrect messages passed along this edge at the k -th iteration when $\mathbf{c}$ is transmitted. Thus, $p^k(\mathbf{c})$ can be expressed as

$$p^k(\mathbf{c}) = \sum_{i=1}^{N_v(k)} p^k(\theta_i) \Pr(\theta_i | \mathbf{c}) \quad (6.12)$$

where $\Pr(\theta_i | \mathbf{c})$ is the probability that the tree is of type θ_i given the transmitted sequence $\mathbf{c}$.

Theorem 6.1: Let $e^k(\mathbf{c})$ be the number of erroneous variable-to-check node messages when $\mathbf{c}$ is the transmitted codeword and k rounds of the message-passing decoding algorithm have been implemented. Assume the code graph is chosen uniformly at random from the ensemble of all graphs. For any $\epsilon > 0$, there exists a positive constant μ , such that if N is sufficiently large, then

$$\Pr \left(\left| \frac{e^k(\mathbf{c})}{N_e} - p^k(\mathbf{c}) \right| > \epsilon \right) \leq 2e^{-\mu\epsilon^2 N}. \quad (6.13)$$

Proof: Please refer to Appendix B. ■

Theorem 6.1 states that for a particular transmitted sequence $\mathbf{c}$, as the code length N tends to infinity, the fraction of incorrect variable-to-check node messages in the decoding converges to the probability $p^k(\mathbf{c})$ which represents the fraction of incorrect messages assuming the message-flow neighborhood is tree-like.

We denote $\mathbf{c}_{\text{i.u.d.}}$ as an i.u.d. random sequence. For a given edge $\vec{e}$ whose message-flow neighborhood of depth k is tree-like, let $p_{\text{i.u.d.}}^k$ be the expected number of incorrect messages passed along this edge at the t -th iteration when an i.u.d. random sequence is transmitted. The value $p_{\text{i.u.d.}}^k$ can be expressed as

$$\begin{aligned}
p_{\text{i.u.d.}}^k &= \mathbf{E} [p^k(\mathbf{c}_{\text{i.u.d.}})] \\
&= \sum_{j=1}^{2^N} 2^{-N} p^k(\mathbf{c}_j) \\
&= \sum_{j=1}^{2^N} 2^{-N} \sum_{i=1}^{N_v(k)} p^k(\theta_i) \Pr(\theta_i | \mathbf{c}_j) \\
&= \sum_{i=1}^{N_v(k)} p^k(\theta_i) \sum_{j=1}^{2^N} 2^{-N} \Pr(\theta_i | \mathbf{c}_j) \\
&= \sum_{i=1}^{N_v(k)} p^k(\theta_i) \Pr(\theta_i | \mathbf{c}_{\text{i.u.d.}}) \\
&= \sum_{i=1}^{N_v(k)} p^k(\theta_i) \frac{1}{N_v(k)}. \tag{6.14}
\end{aligned}$$

The last step of the above equation comes from the fact that all neighborhood types are equally probable when the transmitted sequence is i.u.d. Note that (6.14) just means to show that the expected number of incorrect messages passed along a certain edge at the t -th iteration when an i.u.d. random sequence is transmitted is equal to the average probability that a tree delivers an incorrect message. Thus, we can transmit

i.u.d. random sequences and use the corresponding error rate to evaluate the average system performance. We do not need to calculate the large number of terms in (6.14).

Theorem 6.2: Let $e^k(\mathbf{c}_{\text{i.u.d.}})$ be the number of erroneous variable-to-check messages when an i.u.d. codeword $\mathbf{c}_{\text{i.u.d.}}$ is transmitted and k rounds of the message-passing algorithm have been implemented. Assume the code graph is chosen uniformly at random from the ensemble of all graphs. For any $\epsilon > 0$, there exists a positive constant μ such that if N is sufficiently large, then

$$\Pr \left(\left| \frac{e^k(\mathbf{c}_{\text{i.u.d.}})}{N_e} - p_{\text{i.u.d.}}^k \right| > \epsilon \right) \leq 2e^{-\mu\epsilon^2 N}. \quad (6.15)$$

Proof: Please refer to Appendix C. ■

Theorem 6.2 states that if an i.u.d. random sequence is transmitted, then the probability of a variable-to-check message being incorrect after k rounds of the message-passing decoding algorithm is highly concentrated around the probability $p_{\text{i.u.d.}}^k$ in which the depth- k message-flow neighborhood is tree-like. Thus, we can use $p_{\text{i.u.d.}}^k$ to evaluate the performance of the system with i.u.d. transmitted sequence.

6.2.3 Density Evolution and Threshold Calculation

Based on the tree-like message-flow neighborhood, we can analyze the decoding algorithm by tracking the evolution of the message densities during the iterations. This method was used in [23] to analyze the performance of LDPC codes over memoryless channels, where it was called the density evolution. Let $\tau(x_{i,j})$ be the *a posteriori* LLR value multiplied with the corresponding bit $x_{i,j}$, e.g., $\tau^k(x_{i,j}) = L_{sv}^k(x_{i,j}) \cdot x_{i,j}$. Let f represent the probability density function (pdf) of the corresponding variable. The evolution of the message densities of $\tau^k(x_{i,j})$ through the state-processing node can be

expressed as

$$f_{\tau}^k = \xi_s(f_{L_{vs}}^{k-1}, f_N) \quad (6.16)$$

where $\xi_s(\cdot)$ is symbolic notation of the message density evolution in the state-processing node update as (5.1)-(5.10). Note that $f_{L_{vs}}^{k-1}$ and f_N represent the pdfs of L_{vs}^{k-1} and the channel noise, respectively. Since no closed-form expression of $\xi_s(\cdot)$ can be obtained, it is calculated using Monte Carlo techniques. The evolution of average message densities in the variable-check node update can be expressed as

$$f_z^{l,k} = f_{\tau}^k * \lambda(f_q^{l-1,k}) \quad (6.17)$$

$$f_q^{l,k} = \rho[\xi(f_z^{l,k})] \quad (6.18)$$

and

$$f_{L_{vs}}^k = \bar{\lambda}(f_q^{l,k}) \quad (6.19)$$

where $f_z^{l,k}$, $f_q^{l,k}$, and $f_{L_{vs}}^k$ represent the pdfs of $z_{i,j,m}^{l,k}$, $q_{m,i,j}^{l,k}$, and L_{vs}^k , respectively. The symbolic notation $\xi(\cdot)$ represents the average message density obtained by evolving the density $f_z^{l,k}$ through a check node, and $\bar{\lambda}(x) = \sum_{i=1}^{d_v^{max}} \lambda_i x^i / [i \int_0^1 \lambda(x) dx]$. From (6.14),

the value $p_{\text{i.u.d.}}^k$ can be calculated as

$$\begin{aligned}
p_{\text{i.u.d.}}^k &= \sum_{i=1}^{N_v(k)} p^k(\theta_i) \frac{1}{N_v(k)} \\
&= \sum_{i=1}^{N_v(k)} \frac{1}{N_v(k)} \int_{-\infty}^0 f_{z|\theta_i}^{T_c, k}(\tau|\theta_i) c_{\theta_i} d\tau \\
&= \int_{-\infty}^0 \left[\sum_{i=1}^{N_v(k)} \frac{1}{N_v(k)} f_{z|\theta_i}^{T_c, t}(\tau|\theta_i) c_{\theta_i} \right] d\tau \\
&= \int_{-\infty}^0 f_z^{T_c, t}(\tau) d\tau.
\end{aligned} \tag{6.20}$$

Since the performance of the system degrades monotonously as the standard deviation σ of the AWGN increases for a given channel response matrix, the threshold corresponds to the supremum of all the values of σ such that the fraction of incorrect messages converges to zero as the code length and number of iterations tend to infinity. For the turbo-equalized system in which k rounds of outer iterations are implemented between the state-processing nodes and variable-check nodes and T_c rounds of iterations are implemented between the variable nodes and check nodes per outer iteration, the threshold is defined as

$$\sigma_{te} = \sup \left\{ \sigma : \lim_{N \rightarrow \infty} p_{\text{i.u.d.}}^k \rightarrow 0 \right\}. \tag{6.21}$$

For the non-turbo equalized system in which there is no outer iteration between the state-processing nodes and variable-check nodes but only T_c iterations between the variable nodes and check nodes, the threshold is defined as

$$\sigma_{non-te} = \sup \left\{ \sigma : \lim_{N \rightarrow \infty} \int_{-\infty}^0 f_z^{T_c, 0}(\tau) d\tau \rightarrow 0 \right\}. \tag{6.22}$$

6.3 Numerical Results and Discussions

In this section, we will present some simulation results and the thresholds of the LDPC coded 2-D interference system decoded by the JMPDA. For comparisons, we will also include the simulation results and thresholds of LDPC codes over an AWGN channel [23] and the LDPC coded 2-D interference system detected by a suboptimal strip-wise algorithm (SSWA) [83]. The SSWA is called MAP algorithm in [83] because it is an optimal detection algorithm when only signals of three adjacent tracks are processed to detect the bit sequence in the middle track. For the SSWA, LDPC code is applied to the middle track and iterations are introduced between the detector and the LDPC decoder. Thus, the threshold for the SSWA can be comprehended as the threshold of the LDPC coded 2-D interference system where three tracks are used to detect the middle track. Note that the thresholds of SSWA are obtained by blindly adopting the same method as introduced in the previous section, but the concentration results may not be guaranteed.

For the calculation of thresholds for turbo-equalized system, we adopt a dynamic method. We first fix an error rate step P_{ers} (e.g., $P_{ers} = 10^{-6}$). In the k -th outer iteration, the iterations between the variable nodes and check nodes will proceed until the performance improvement achieved by the next iteration is smaller than the error rate step, i.e.,

$$\int_{-\infty}^0 f_z^{l+1,k}(\tau) d\tau - \int_{-\infty}^0 f_z^{l,k}(\tau) d\tau < P_{ers}. \quad (6.23)$$

Once (6.23) is satisfied, the iterations between the variable nodes and check nodes cease and the system moves to the $(k+1)$ -th outer iteration. This dynamic method can save some computations while generating almost identical results in the threshold

Table 6.1. Evolution of $p_{\text{i.u.d.}}^k$ during the iterations for various parameters.

k	channel $\mathbf{H}_A$		channel $\mathbf{H}_B$	
	$(3,6), \sigma = 0.81$	$(4,36), \sigma = 0.45$	$(3,6), \sigma = 0.84$	$(4,36), \sigma = 0.47$
1	0.120135	0.024950	0.113098	0.022694
2	0.101857	0.019691	0.097696	0.019235
3	0.086699	0.013585	0.088068	0.015846
4	0.011226	0.000000	0.063708	0.013504
5	0.000000	0.000000	0.000000	0.000000

Table 6.2. Threshold values for regular LDPC codes over AWGN and 2-D ISI channels.

(d_v, d_c)	R	AWGN	2D ISI: $\mathbf{H}_A$			2D ISI: $\mathbf{H}_B$		
		σ_{AWGN}	σ_{te}	σ_{non-te}	σ_{SSWA}	σ_{te}	σ_{non-te}	σ_{SSWA}
(3, 4)	0.25	1.26	1.19	1.10	1.12	1.22	1.15	1.14
(3, 6)	0.50	0.88	0.81	0.73	0.74	0.84	0.78	0.75
(3, 10)	0.70	0.68	0.62	0.56	0.54	0.64	0.59	0.53
(3, 20)	0.85	0.54	0.50	0.46	0.39	0.52	0.49	0.38
(4, 36)	0.89	0.50	0.45	0.43	0.35	0.47	0.46	0.33

calculation. Since it has been shown in Chapter 5 that there is hardly any performance improvement by introducing more iterations when $T_s > 3$, we fix $T_s = 3$ for the subsequent analysis.

Table 6.1 shows the evolution of $p_{\text{i.u.d.}}^k$ during the iterations for various parameters, namely, channel, degree distribution, and standard deviation of the noise. The results are obtained using regular LDPC codes. We denote the degree distribution as (d_v, d_c) . Thus, the code rate is $1 - d_v/d_c$. It can be observed that for the given parameters, the error probability $p_{\text{i.u.d.}}^k$ becomes smaller and smaller during the iterations and an error probability smaller than 10^{-6} can be achieved within 5 iterations. However, if

the standard deviation σ exceeds the threshold, the error probability will remain at a certain value and will not evolve as the iteration process proceeds. Table 6.2 shows the thresholds of the LDPC coded 2-D interference system decoded by the JMPDA and SSWA as well as thresholds of LDPC codes over an AWGN channel. We denote the threshold of the LDPC code over an AWGN channel and SSWA as σ_{AWGN} and σ_{SSWA} , respectively in Table 6.2. For the 2-D interference channel, the threshold is normalized by a factor $|\mathbf{H}| = |\sum_{i=0}^{M_h-1} \sum_{j=0}^{N_h-1} (h(i, j))^2|^{1/2}$. If the standard deviation σ is below the calculated threshold value, then any arbitrarily small target error probability can be achieved by using a sufficiently long code with sufficient number of iterations. It can be observed that as the code rate increases, the threshold decreases. Since the interference level of channel $\mathbf{H}_B$ is smaller than that of $\mathbf{H}_A$, the threshold of channel $\mathbf{H}_B$ is larger than that of $\mathbf{H}_A$ when the system is decoded by the JMPDA. The JMPDA always has a larger threshold than that of the SSWA and the performance gap becomes larger as the code rate increases. This is because the SSWA only processes three tracks to detect the data sequence in the middle track, while the JMPDA detects the data in two directions and processes the information from more than three tracks.

Next, we examine how tight the threshold upper bounds the performance of the system. Since the results are similar for both channels $\mathbf{H}_A$ and $\mathbf{H}_B$, we only adopt the channel $\mathbf{H}_A$ in the following. Note that here the thresholds correspond to the infimum of SNR such that the message error rate converges to zero as the number of iterations tends to infinity. For a fixed 2-D channel response matrix, the threshold corresponding to σ and the threshold corresponding to SNR are one-to-one mapping. The LDPC codes adopted in the simulation are constructed using the semi-random algorithm [19]. We denote the LDPC codes as (N, K) . There are two sets of LDPC codes used in our simulation. The first set contains LDPC codes $(14000, 7000)$, $(6000, 3000)$ and $(2000, 1000)$, which are all regular LDPC codes with code rate $R = 0.5$, variable node

Figure 6.4. Simulation results and thresholds for turbo-equalized and non-turbo equalized JMPDA with different LDPC codes and different numbers of iterations.

degree $d_v = 3$ and check node degree $d_c = 6$. The second set contains LDPC codes $(18000, 16000)$, $(9000, 8000)$ and $(4500, 4000)$, which are all regular LDPC codes with code rate $R = 0.89$, variable node degree $d_v = 4$ and check node degree $d_c = 36$. For the simulation, we let the coset-defining vector $\mathbf{b} = \mathbf{0}$.

Figure 6.4 shows the simulation results and the thresholds of turbo-equalized and non-turbo equalized 2-D interference systems decoded by the JMPDA. The simulation results are based on two LDPC codes, namely, $(14000, 7000)$ and $(18000, 16000)$. We denote the number of iterations as T_c/T_o in the figure. Note that $T_o = 0$ corresponds to non-turbo equalized system and $T_o > 0$ corresponds to the system with turbo-equalization. It can be observed that as the number of iterations increases, the simulation results approach closer to the corresponding thresholds. The effect of iterations is

Figure 6.5. Simulation results and thresholds for the LDPC codes with $d_v = 3$, $d_c = 6$ and $R = 0.5$.

more significant for code (14000, 7000) as compared to that of code (18000, 16000).

Figures 6.5 and 6.6 show the simulation results and the corresponding thresholds for the first and second set of LDPC codes, respectively. We compare the results of LDPC codes over an AWGN channel, and the LDPC coded 2-D interference channel decoded by the JMPDA and SSWA. In these figures, the solid curves correspond to the LDPC codes over an AWGN channel. The dashed curves correspond to the LDPC coded 2-D interference system decoded by the JMPDA with turbo equalization. The dotted curves correspond to the LDPC coded 2-D interference system detected by SSWA. For the AWGN channel, the SNR is defined as

Figure 6.6. Simulation results and thresholds for the LDPC codes with $d_v = 4$, $d_c = 36$ and $R = 0.89$.

$$\text{SNR} = 10 \log_{10} \left(\frac{1}{2R\sigma^2} \right) \text{ dB}. \quad (6.24)$$

In these simulation results, the number of iterations between the variable nodes and check nodes is set to $T_c = 50$ and the number of outer iterations between the state-processing nodes and the variable-check nodes is fixed at $T_o = 10$. In Fig. 6.5, for the three system setups, the simulation results with code (14000,7000) are about 0.4 dB away from the corresponding threshold. The simulation results and thresholds of JMPDA are about 0.8 dB worse than those of LDPC code over an AWGN channel and are about 0.7 dB better than those of SSWA. In Fig. 6.6, the performance gaps

between the thresholds and simulation results with code (18000, 16000) are less than 0.4 dB for all three system setups. This time, as the code rate increases, the simulation results and thresholds of JIDDS are about 0.9 dB worse than those of LDPC codes over an AWGN channel and are about 2.2 dB better than those of SSWA. Note that for the JMPDA with the settings of $T_c = 50$ and $T_o = 10$, the actual total number of iterations between the variable nodes and check nodes is $T_c \cdot T_o = 500$. On the other hand, for the LDPC codes over an AWGN channel, the total number of iterations between the variable nodes and check nodes is $T_c = 50$. Thus, due to the neighborhoods dependence in the channel, more iterations are needed than that of the AWGN for the JMPDA to approach the threshold. It can be observed in these two figures that as the code length N increases, the simulation results approach the thresholds. When code length and number of iterations increase, the simulation results are expected to approach even closer to the thresholds. The small gaps between the simulation results and the corresponding thresholds confirm the effectiveness of the density evolution technique in analyzing the performance of the system for large code lengths.

6.4 Conclusion

In this chapter, we have analyzed the performance of LDPC codes over 2-D interference channels. The JMPDA is adopted in the 2-D interference system. We have shown the graph representation of the message-flow neighborhood of the JMPDA and proved that the message-flow neighborhood will be tree-like for a long code length. We have also shown two concentration theorems. The two concentration theorems state that for a particular transmitted codeword or i.i.d. random codeword, the message error probability of the system concentrates around the probability in which the message-flow neighborhood is tree-like. For the tree-like message-flow neighborhood, a modified den-

sity evolution algorithm has been employed to track the message densities during the iterations. We have also used the density evolution algorithm to calculate the threshold values of the JMPDA and SSWA. For a fixed channel response matrix, the threshold represents the largest noise level the system can tolerate in order to achieve reliable transmission. Thus, whenever the standard deviation of noise is below the threshold, transmission will be reliable given that the code length and number of iterations are large enough. The simulation results have been compared to the corresponding thresholds and the gaps between them are relatively small. Therefore, the threshold calculated using the modified density evolution can be used to accurately predict the system performance for large code lengths.

Chapter 7

Conclusions and Future Work

7.1 Concluding Remarks

The problem of 2-D interference arises with the detection of next-generation ultra-high-density magnetic recording systems. Unlike the conventional 1-D interference channel in which the interference comes from one direction, in 2-D interference channels, the interference exists in two orthogonal directions. Thus, the traditional Viterbi algorithm or BCJR algorithm cannot be applied to the detection of 2-D interference channels.

We have studied the maximum-likelihood page detection (MLPD) for 2-D interference channels. The MLPD is the optimal detection algorithm for 2-D interference channels but it is difficult to be implemented in practice due to its high computational complexity. We have derived the performance bound of MLPD which provides an optimal benchmark for the detection of 2-D interference channels. It has been shown that the performance of MLPD does not degrade monotonically as the channel interference level increases. We have analyzed the error patterns of the MLPD with respect to the interference level and explained the non-monotonic observation.

Next, we have proposed a suboptimal detection scheme called the joint iterative

detection/decoding scheme (JIDDS). In JIDDS, the channel detector consists of two constituent detectors, namely, the down-track detector and cross-track detector, which extend the BCJR algorithm to two directions. Low-density parity-check (LDPC) codes have been applied to the JIDDS to further improve the system performance. Several iteration schemes can be introduced in the JIDDS. The computational complexity of the system has been analyzed and we have proposed a reduced-state detection algorithm to greatly reduce the system complexity at the cost of marginal performance loss. We have investigated the effects of 2-D array size, channel interference level, and number of iterations on the system performance via simulations. We have also compared the performance and complexity of the JIDDS with other algorithms. It has been shown that the proposed JIDDS can achieve near optimal performance and outperform other suboptimal algorithms, such as the suboptimal strip-wise algorithm (SSWA) and the Markov chain Monte Carlo (MCMC) algorithm.

In order to decode the LDPC coded 2-D interference system efficiently, we have proposed a joint message-passing decoding algorithm (JMPDA). The JMPDA consists of state-processing node update and variable-check node update which can be implemented together in parallel. The JMPDA can be viewed as a windowed version of the JIDDS. We have also used extrinsic information transfer (EXIT) chart to analyze the convergence behavior of the JMPDA. By analyzing the individual EXIT curve, the effects of various parameters on the performance of the JMPDA has been studied.

Lastly, we have explained the message-flow neighborhood of the JMPDA and proved that the message-flow neighborhood will be tree-like given a sufficiently long code. We have shown two concentration theorems that the average error rate of the JMPDA converges to the value which is obtained assuming the message-flow neighborhood is tree-like. We have used a modified density evolution (DE) algorithm to track the evolution of probability density during the iterations. Following that, we have also

computed the threshold values which represent the performance limit of the JMPDA. Simulation results with long code lengths and large numbers of iterations have been compared to the threshold values to show the validity of the modified DE in analyzing the system performance.

7.2 Future Work

In the following, we will present some topics for future research work.

LDPC code is a 1-D sequence but the recorded bits form a 2-D array. In order to map the LDPC coded 1-D sequence to a 2-D array, a 1-D to 2-D mapping scheme is required. There are some straight-forward 1-D to 2-D mapping schemes, e.g. block mapping scheme and random mapping scheme. A block interleaver is a mapping scheme that distributes the 1-D sequence $\mathbf{c} = \{c_i, i = 1, 2, \dots, N\}$ to a 2-D array $\mathbf{x} = \{x_{j,k}, j = 1, 2, \dots, N_r, k = 1, 2, \dots, N_c\}$ such that $x_{j,k} = c_i$ with $j = \lfloor i/N_c \rfloor$ and $k = i - (j - 1) \times N_c$. A random interleaver maps the 1-D sequence randomly on the 2-D array. It can be implemented as a random 1-D interleaver which randomly redistributes the 1-D sequence $\mathbf{c}$ to a new sequence $\mathbf{c}' = \{c'_i, i = 1, 2, \dots, N\}$ and then followed by a block interleaver which maps the new sequence $\mathbf{c}'$ to a 2-D array $\mathbf{x}$.

However, these mapping schemes may result in short cycles in the joint detection and decoding. Figure 7.1 illustrates this problem. In this figure, the dark circles represent the check nodes and the dark grids represent the variable nodes connected to those check nodes. The grids adjacent to the bit $x_{i,j}$ form the interference region of $x_{i,j}$ which is represented by grids with crosses and they can directly interfere with $x_{i,j}$. Thus, in the channel detection, the soft estimation of $x_{i,j}$ is directly related to the information of bits in its interference region. If the bits within one interference region are connected to the same check node as the case in Fig. 7.1, then short cycles will be

Figure 7.1. 1-D to 2-D mapping scheme that results in short cycles

generated in the joint detection and decoding scheme, which will be very harmful to the system performance. Figure 4.2 shows a preferred case in which the variable nodes are separated far away from each other. The ideal mapping scheme is to ensure that every variable node connected to the same check node lies a large distance away with each other. The 1-D to 2-D mapping scheme can affect the performance of the joint detection/decoding system. A well designed 1-D to 2-D mapping scheme can make the message-flow neighborhood of the JMPDA be tree-like which is the key assumption used in Chapter 6 and it is expected to improve the performance of the JIDDS and JMPDA. In order to demonstrate this point, we will introduce the following experiment.

In the previous chapters, the input sequence is just one LDPC codeword $\mathbf{c} = \{c_1, c_2, \dots, c_N\}$ with length N . Here we let the input sequence be N_w LDPC codewords

Figure 7.2. Block diagram of the experimental system.

in a row. The N_w LDPC codewords are all the same with length N , i.e., $\{\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{N_w}\}$ with $\mathbf{c}_i = \{c_1, c_2, \dots, c_N\}$, as shown in Fig. 7.2. A random interleaver is adopted and the other parts of the system remain the same. Thus, the only difference of the new system is the input sequence length and the corresponding interleaver size become larger. Figure 7.3 shows the simulation results of the JMPDA for $N_w = 6$, $N_w = 3$ and $N_w = 1$ based on channel $\mathbf{H}_A$. Here, we use the same LDPC code (1998, 222) and set the numbers of iterations as $T_c = 50$, $T_s = 3$, and $T_o = 5$. It can be observed that a larger N_w can significantly improve the system performance. Since it has been shown in Chapter 4 that the size of the 2-D array has no effect on the system performance, the only explanation for this performance gain is that a larger N_w can lead to a better interleaver. As we adopt a random interleaver, a larger N_w can decrease the chance that the bits within one interference region are connected to the same check node, and hence improve the system performance. Therefore, it is desirable to propose some 1-D to 2-D mapping scheme design methods to further improve the system performance in the future work.

Secondly, in this thesis, we have investigated the application of LDPC codes to 2-D ISI channels. But note that in Fig. 5.5 the near abrupt behavior of the powerful LDPC code is not well matched to the channel detector. We may borrow some ideas from [104]

Figure 7.3. Performance comparison of systems with different N_w .

to find some error-correcting codes (ECCs) which can be well matched to the channel detector and further improve the system performance. Other than LDPC codes, there are also some well-performing ECCs, such as the non-binary LDPC codes [105],[106] and turbo product codes (TPCs) [107],[108]. Non-binary LDPC codes have been shown to possess a stronger error-correcting capability than binary LDPC codes[105]. The application of non-binary LDPC codes to 2-D ISI channels has been studied in [91]. However, in this model, due to the channel symmetry problem, the LDPC codes must reduce the symbol alphabet in order to fit the channel detector. This results in severe code rate loss. Thus, it is desirable to propose other methods to apply non-binary LDPC codes to 2-D ISI channels. TPCs exhibit excellent error performance and have many appealing properties for magnetic recording systems, such as high code rate,

linear encoding/decoding complexity, and parallelizable implementation. Like the 2-D recorded data in 2-D ISI systems, TPCs also work on a 2-D array. Thus, the study of the application of TPCs to 2-D ISI channels is a promising topic for future research work.

In JIDDS, our proposed channel detector adopts trellis-based sequence detection algorithms in two dimensions to produce the likelihoods. But the performance gain achieved by the complex trellis machinery is limited as shown in Fig. 5.6. However, as shown in Figures 4.2 and 6.1, in 2-D ISI channels, each bit actually has a sphere of influence around it. Thus, the trellis-based method may not be the most efficient way of detection over 2-D ISI channels. It might be promising to design some detection algorithm which computes the center bit posterior probability by marginalizing over that neighborhood directly. There are precedencies for this effect and approach [109]. Besides, the capacity of 2-D ISI channels can provide an ultimate benchmark to evaluate the detector and decoder over 2-D ISI channels. However, the actual capacity of 2-D ISI channels is still an open problem so far and this could be addressed in future work.

In this thesis, we only focus on the signal processing of HDDs which serves as only one of the many factors on the data rate of HDDs. The practical data rate of an HDD is dependent on multiple factors, such as the spin speed of the disk, the recording density of the disk, the interfaces, the signal processing speed of the circuit, etc. A more complete HDD and the corresponding data rate issue can be considered as future work.

Author's Publications

Journal Publications

1. J. Yao, K. C. Teh, and K. H. Li, "Reduced-state Bahl-Cocke-Jelinek-Raviv detector for patterned media storage," *IEEE Trans. Magn.*, vol. 46, no. 12, pp. 4108-4110, Dec. 2010.
2. J. Yao, K. C. Teh, and K. H. Li, "Performance evaluation of maximum-likelihood page detection for 2D interference channels," *IEEE Trans. Magn.*, vol. 48, no. 7, pp. 2239-2242, July 2012.
3. J. Yao, K. C. Teh, and K. H. Li, "Joint iterative detection/decoding scheme for discrete two-dimensional interference channels," *IEEE Trans. Commun.*, vol. 60, no. 12, pp. 3548-3555, Dec. 2012.
4. J. Yao, K. C. Teh, and K. H. Li, "Joint message-passing decoding of LDPC codes and 2D ISI channels," *IEEE Trans. Magn.*, vol. 49, no. 2, pp. 675-681, Feb. 2013.

Conference Publications

1. J. Yao, K. C. Teh, and K. H. Li, "Joint message-passing decoding of LDPC codes and 2D ISI channels," in the 23rd Magnetic Recording Conference (TMRC 2012).
2. J. Yao, K. C. Teh, and K. H. Li, "Iterative detection scheme with LDPC codes for two-dimensional interference channels," in *Proc. IEEE Int. Conf. Commun. Systems (ICCS)*, Nov. 2012.

Bibliography

- [1] E. D. Daniel, C. D. Mee, and M. H. Clark, *Magnetic Recording: The First 100 Years*, IEEE Press, Piscataway, 1999.
- [2] B. Vasic and E. M. Kurtas, *Coding and Signal Processing for Magnetic Recording Systems*, CRC Press, 2005.
- [3] S. Lin and D. J. Costello, *Error Control Coding*, 2nd ed. New Jersey: Pearson Education, Inc., 2004.
- [4] M. H. Kryder, “Magnetic recording beyond the superparamagnetic limit,” in *Proc. IEEE Int. Magn. Conf. (INTERMAG)*, 2000, pp. 575.
- [5] “Hitachi achieves nanotechnology milestone for quadrupling terabyte hard drive” (Press release), Hitachi, Oct. 15, 2007. Retrieved Sep. 2011.
- [6] R. L. White, R. M. H. New, and R. F. W. Pease, “Patterned media: A viable route to 50 Gbit/in² and up for magnetic recording,” *IEEE Trans. Magn.*, vol. 33, no. 1, pp. 990-995, Jan. 1997.
- [7] R. E. Rottmayer, S. Batra, D. Buechel, W. A. Challener, J. Hohlfield, Y. Kubota, L. Li, B. Lu, C. Mihalcea, K. Mountfield, K. Pelhos, C. Peng, T. Rausch, M. A. Seigler, D. Weller, and X. Yang, “Heat-assisted magnetic recording,” *IEEE Trans. Magn.*, vol. 42, no. 10, pp. 2417-2421, Oct. 2006.

- [8] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 Terabits per square inch on conventional media," *IEEE Trans. Magn.*, vol. 45, no. 2, pp. 917-923, Feb. 2009.
- [9] P. W. Nutter, I. T. Ntokas, and B. K. Middleton, "An investigation of the effects of media characteristics on read channel performance for patterned media storage," *IEEE Trans. Magn.*, vol. 41, no. 11, pp. 4327-4334, Nov. 2005.
- [10] Z. Zhang, T. M. Duman, and E. M. Kurtas, "Achievable information rates and coding for MIMO systems over ISI channels and frequency-selective fading channels," *IEEE Trans. Commun.*, vol. 52, no. 10, pp. 1698-1710, Oct. 2004.
- [11] R. Zakaria and D. L. Ruyet, "On maximum likelihood MIMO detection in QAM-FBMC systems," in *Proc. IEEE Int. Symp. on Personal, Indoor and Mobile Radio Communications (PIMRC)*, 2010, pp. 183-197.
- [12] A. Dholakia, E. Eleftheriou, T. Mittelholzer, and M. P. C. Fossorier, "Capacity-approaching codes: Can they be applied to the magnetic recording channel?," *IEEE Commun. Mag.*, vol. 42, no. 2, pp. 122-130, Feb. 2004.
- [13] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 498-519, Feb. 2001.
- [14] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, May 1993, pp. 1064-1070.
- [15] C. Berrou and A. Glavieux, "Near optimum error correcting coding and decoding: Turbo-codes," *IEEE Trans. Commun.*, vol. 44, no. 10, pp. 1261-1271, Oct. 1996.

- [16] S. ten Brink, "Rate one-half code for approaching the Shannon limit by 0.1 dB," *Electronics Letter*, vol. 36, no. 15, pp. 1293-1294, July 2000.
- [17] R. G. Gallager, *Low-Density Parity-Check Codes*. Cambridge, MA: MIT Press, 1963.
- [18] D. J. C. MacKay and R. M. Neal, "Near Shannon limit performance of low-density parity-check codes," *Electron. Lett.*, vol. 32, no. 18, pp. 1645-1646, Aug. 1996.
- [19] 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.
- [20] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1732-1736, Nov. 1996.
- [21] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs and belief propagation," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Cambridge, MA, USA, Aug. 1998, pp. 117
- [22] M. G. Luby, M. Mitzenmacher, M. A. Shokrollahi, and D. A. Spielman, "Improved low-density parity-check codes using irregular graphs," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 585-598, Feb. 2001.
- [23] T. J. Richardson and R. L. 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.
- [24] S. -Y. Chung, G. D. Forney, Jr., T. J. 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.

- [25] S. -Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 657-670, Feb. 2001.
- [26] T. J. Richardson, A. Shokrollahi and R. L. Urbanke, "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Trans. Inform. Theory*, vol. 47, no. 2, pp. 619-637, Feb. 2001.
- [27] M. Yang, W. E. Ryan, and Y. Li, "Design of efficiently encodable moderate-length high-rate irregular LDPC codes," *IEEE Trans. Commun.*, vol. 52, no. 4, pp. 564-571, April 2004.
- [28] 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, pp. 2711-2736, Nov. 2001.
- [29] G. Liva, S. Song, L. Lan, Y. Zhang, W. Ryan, and S. Lin, "Desin of LDPC codes: A survey and new results," *J. Comm. Software Syst.*, vol. 2, no. 3, pp. 191-211, Sep. 2006.
- [30] W. E. Ryan, "An introduction to LDPC codes," in *CRC Handbook for Coding and Signal Processing for Recording Systems (B. Vasic, ed.)*, CRC Press, 2004.
- [31] R. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inform. Theory*, vol. 27, no. 5, pp. 533-547, Sep. 1981.
- [32] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, pp. 268-278, Mar. 1973.

- [33] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. 20, pp. 284-287, Mar. 1974.
- [34] G. D. Forney, "Lower bounds on error probability in the presence of large intersymbol interference," *IEEE Trans. Commun.*, vol. 20, no. 1, pp. 76-77, Feb. 1972.
- [35] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, no. 3, pp. 363-378, May 1972.
- [36] S. Verdu, "Maximum likelihood sequence detection for intersymbol interference channels: A new upper bound on error probability," *IEEE Trans. Inform. Theory*, vol. 33, no. 1, pp. 62-68, Jan. 1987.
- [37] S. A. Raghavan, J. K. Wolf, and L. B. Milstein, "On the performance evaluation of ISI channels," *IEEE Trans. Inform. Theory*, vol. 39, no. 3, pp. 957-965, May 1993.
- [38] W. Hirt and J. L. Massey, "Capacity of the discrete-time Gaussian channel with intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 34, no. 3, pp. 380-388, May 1988.
- [39] S. Shamai, L. H. Ozarow, and A. D. Wyner, "Information rates for a discrete-time Gaussian channel with intersymbol interference and stationary inputs," *IEEE Trans. Inform. Theory*, vol. 37, no. 6, pp. 1527-1539, Nov. 1991.

- [40] S. Shamai and R. Laroia, "The intersymbol interference channel: Lower bounds on capacity and channel precoding loss," *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1388-1404, Sep. 1996.
- [41] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo-equalization," *Eur. Trans. Telecommun.*, vol. 6, pp. 507-511, Sep.-Oct. 1995.
- [42] D. Raphaeli and Y. Zarai, "Combined turbo equalization and turbo decoding," *IEEE Commun. Lett.*, vol. 2, pp. 107-109, Apr. 1998.
- [43] R. Koetter, A. C. Singer, and M. Tuchler, "Turbo equalization," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 67-80, Jan. 2004.
- [44] M. Tuchler and A. C. Singer, "Turbo equalization: An overview," *IEEE Trans. Inform. Theory*, vol. 57, no. 2, pp. 920-952, Feb. 2011.
- [45] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecomm. Conf. (GLOBECOM)*, vol. 3, Nov. 1989, pp. 1680-1686.
- [46] T. Mittelholzer, A. Dholakia, and E. Eleftheriou, "Reduced-complexity decoding of low density parity check codes for generalized partial response channels," *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 721-728, Mar. 2001.
- [47] M. Yang, and W. E. Ryan, "Performance of efficiently encodable low-density parity-check codes in noise bursts on the EPR4 channel," *IEEE Trans. Magn.*, vol. 40, no. 2, pp. 507-512, Mar. 2004.

- [48] B. M. Kurkoski, P. H. Siegel, and J. K. Wolf, "Joint message-passing decoding of LDPC codes and partial-response channels," *IEEE Trans. Inform. Theory*, vol. 48, no. 6, pp. 1410-1422, June 2002.
- [49] H. Song and B. V. K. V. Kumar, "Low-density parity-check codes for partial-response channels," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 56-66, Jan. 2004.
- [50] M. H. Taghavi and P. H. Siegel, "Graph-based decoding in the presence of ISI," *IEEE Trans. Inform. Theory*, vol. 57, no. 4, pp. 2188-2202, Apr. 2011.
- [51] B. H. Kim and H. D. Pfister, "Joint decoding of LDPC codes and finite-state channels via linear-programming," *IEEE J. Select. Topics in Signal Process.*, vol. 5, no. 8, pp. 1563-1576, Dec. 2011.
- [52] H. D. Pfister and P. H. Siegel, "Joint iterative decoding of LDPC codes for channels with memory and erasure noise," *IEEE J. Select. Areas Commun.*, vol. 26, no. 2, pp. 320-337, Feb. 2008.
- [53] R. Radhakrishnan and B. Vasic, "Joint message-passing symbol-decoding of LDPC coded signals over partial-response channels," in *Proc. IEEE Int. Conf. on Commun. (ICC)*, 2009, pp. 1-5.
- [54] P. Pakzad and V. Anantharam, "Kikuchi approximation method for joint decoding of LDPC codes and partial-response channels," *IEEE Trans. Commun.*, vol. 54, no. 7, pp. 1149-1153, July 2006.
- [55] G. Colavolpe and G. Germini, "On the application of factor graphs and the sum-product algorithm to ISI channels," *IEEE Trans. Commun.*, vol. 53, no. 5, pp. 818-825, May 2005.

- [56] G. Colavolpe, "On LDPC codes over channels with memory," *IEEE Trans. Wireless Commun.*, vol. 5, no. 7, pp. 1757-1766, July 2006.
- [57] S.-J. Lee, A. C. Singer, and N. R. Shanbhag, "Linear turbo equalization analysis via BER transfer and EXIT charts," *IEEE Trans. Signal Process.*, vol. 53, no. 8, pp. 2883-2897, Aug. 2005.
- [58] M.-K. Lee and K. Yang, "Scheduling for an adaptive number of iterations in turbo equalizers combined with LDPC decoders," *IEEE Trans. Commun.*, vol. 58, no. 10, pp. 2759-2764, Oct. 2010.
- [59] P. Tan and J. Li, "Finite-length extrinsic information transfer (EXIT) analysis for coded and precoded ISI channels," *IEEE Trans. Magn.*, vol. 44, no. 5, pp. 648-655, May 2008.
- [60] T. M. Duman and E. M. Kurtas, "Performance bounds for high rate linear codes over partial-response channels," *IEEE Trans. Inform. Theory*, vol. 47, no. 3, pp. 1201-1205, Mar. 2001.
- [61] W. Tan, R. M. Todd, and J. R. Cruz, "Bounds for low-density parity-check codes over partial response channels," *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2310-2312, Sep. 2002.
- [62] A. Kavcic, X. Ma, and M. Mitzenmacher, "Binary intersymbol interference channels: Gallager codes, density evolution, and code performance bounds," *IEEE Trans. Inform. Theory*, vol. 49, no. 7, pp. 1636-1652, July 2003.
- [63] W. Tan and J. R. Cruz, "Performance evaluation of low-density parity-check codes on partial-response channels using density evolution," *IEEE Trans. Commun.*, vol. 52, no. 8, pp. 1253-1256, Aug. 2004.

- [64] A. Thangaraj and S. W. McLaughlin, "Thresholds and scheduling for LDPC-coded partial response channels," *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2307-2309, Sep. 2002.
- [65] S. Nabavi, B. V. K. V. Kumar, and J. A. Bain, "Two-dimensional pulse response and media noise modeling for bit-patterned media," *IEEE Trans. Magn.*, vol. 44, no. 11, pp. 3789-3792, Nov. 2012.
- [66] S. Karakulak, P. H. Siegel, J. K. Wolf, and H. N. Bertram, "A new read channel model for patterned media storage," *IEEE Trans. Magn.*, vol. 44, no. 1, pp. 193-197, Jan. 2008.
- [67] E. Hwang, R. Negi, and B. V. K. V. Kumar, "Signal processing for near 10 Tbit/in² density in two-dimensional magnetic recording (TDMR)," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1813-1816, June 2010.
- [68] Y. Ng, K. Cai, B. V. K. V. Kumar, T. C. Chong, S. Zhang, and B. J. Chen, "Channel modeling and equalizer design for staggered islands bit-patterned media recording," *IEEE Trans. Magn.*, vol. 48, no. 6, pp. 1976-1983, June 2012.
- [69] S. Zhang, K. S. Chai, K. Cai, B. Chen, Z. Qin, and S. M. Foo, "Write failure analysis for bit-patterned-media recording and its impact on read channel modeling," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1363-1365, June 2010.
- [70] M. Yamashita, H. Osawa, Y. Okamoto, Y. Nakamura, Y. Suzuki, K. Miura, and H. Muraoka, "Read/write channel modeling and two-dimensional neural network equalization for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 47, no. 10, pp. 3558-3561, Oct. 2011.

- [71] K. Cai, Z. Qin, S. Zhang, Y. Ng, K. S. Chai, and R. Radhakrishnan, "Modeling, detection, and LDPC codes for bit-patterned media recording," in *IEEE Global Conf. Commun. (GLOBECOM)*, 2010, pp. 1910-1914.
- [72] D. M. Arnold, H.-A. Loeliger, P. O. Vontobel, A. Kavcic, and W. Zeng, "Simulation-based computation of information rates for channels with memory," *IEEE Trans. Inform. Theory*, vol. 52, no. 8, pp. 3498-3508, Aug. 2006.
- [73] J. Chen and P. H. Siegel, "On the symmetric information rate of two-dimensional finite-state ISI channels," *IEEE Trans. Inform. Theory*, vol. 52, no. 1, pp. 227-236, Jan. 2006.
- [74] J. Chen and P. H. Siegel, "Information rates of two-dimensional finite state ISI channels," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, June-July 2003, p. 118.
- [75] G. Shental, N. Shental, and S. Shamai, "On the achievable information rates of finite-state input two-dimensional channels with memory," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, Sep. 2005, pp. 2354-2358.
- [76] M. Marrow and J. K. Wolf, "Iterative detection of 2-dimensional ISI channels," in *Proc. IEEE Inform. Theory Workshop (ITW)*, Paris, France, Mar. 2003, pp. 131-134.
- [77] E. Ordentlich and R. M. Roth, "On the computational complexity of 2D maximum-likelihood sequence detection," HP Labs, 2006, Tech. Rep. HPL-2006-69.

- [78] S. Nabavi, B. V. K. V. Kumar, and J.-G. Zhu, "Modifying Viterbi Algorithm to mitigate intertrack interference in bit-patterned media," *IEEE Trans. Magn.*, vol. 43, no. 6, pp. 2274-2276, June 2007.
- [79] S. Nabavi and B. V. K. V. Kumar, "Two-dimensional generalized partial response equalizer for bit-patterned media," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2007, pp. 6249-6254.
- [80] S. Nabavi, S. Jeon, and B. V. K. V. Kumar, "An analytical approach for performance evaluation of bit-patterned media channels," *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 135-142, Feb. 2010.
- [81] M. Keskinöz, "Two-dimensional equalization/detection for patterned media storage," *IEEE Trans. Magn.*, vol. 44, no. 4, pp. 533-539, Apr. 2008.
- [82] M. Keskinöz, "Soft feedback equalization for patterned media storage," *IEEE Trans. Magn.*, vol. 44, no. 11, pp. 3793-3796, Nov. 2008.
- [83] S. Karakulak, P. H. Siegel, J. K. Wolf, and H. N. Bertram, "Joint-track equalization and detection for bit patterned media recording," *IEEE Trans. Magn.*, vol. 46, no. 9, pp. 3639-3647, Sep. 2010.
- [84] Y. Wu, J. A. O' Sullivan, N. Singla, and R. S. Indeck, "Iterative detection and decoding for separable two-dimensional intersymbol interference," *IEEE Trans. Magn.*, vol. 39, no. 4, pp. 2115-2120, July 2003.
- [85] J. Kim and J. Lee, "Iterative two-dimensional soft output Viterbi algorithm for patterned media," *IEEE Trans. Magn.*, vol. 47, no. 3, pp. 594-597, Mar. 2011.

- [86] T. Cheng, B. J. Belzer, and K. Sivakumar, "Row-column soft-decision feedback algorithm for two-dimensional intersymbol interference," *IEEE Signal Process. Lett.*, vol. 14, no. 7, pp. 433-436, July 2007.
- [87] Y. Zhu, T. Cheng, K. Sivakumar, and B. J. Belzer, "Markov random field detection on two-dimensional intersymbol interference channels," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 2639-2648, July 2008.
- [88] N. Singla, J. A. O' Sullivan, C. T. Miller, and R. S. Indeck, "Decoding for magnetic recording media with overlapping tracks," *IEEE Trans. Magn.*, vol. 41, no. 10, pp. 2968-2970, Oct. 2005.
- [89] N. Singla and J. A. O' Sullivan, "Minimum mean square error equalization using priors for two-dimensional intersymbol interference," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, June-July 2004, p. 131.
- [90] N. Singla and J. A. O' Sullivan, "Joint equalization and decoding for nonlinear two-dimensional intersymbol interference channels," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Adelaide, Australia, Sep. 2005, pp. 1353-1357.
- [91] X. Liu, C. Shi, M. Teng, and X. Ma, "Error correction coding with LDPC codes for patterned media storage," *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 3745-3748, Oct. 2009.
- [92] M. Shaghaghi, K. Cai, Y. L. Guan, and Z. L. Qin, "Markov chain Monte Carlo based detection for two-dimensional intersymbol interference channels," *IEEE Trans. Magn.*, vol. 47, no. 2, pp. 471-478, Feb. 2011.

- [93] L. M. M. Myint, P. Supnithi, and P. Tantaswadi, "An inter-track interference mitigation technique using partial ITI estimation in patterned media storage," *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 3691-3694, Oct. 2009.
- [94] K. M. Chugg, "Performance of optimal digital page detection in a two-dimensional ISI/AWGN channel," in *Proc. Asilomar Conf. Signals, Systems and Comp.*, Pacific Grove, CA, Nov. 1996, vol. 2, pp. 958-962.
- [95] I. Demirkan, P. H. Siegel, and J. K. Wolf, "Error event characterization on 2-D ISI channels," *IEEE Trans. Inform. Theory*, vol. 55, no. 3, pp. 1146-1152, Mar. 2009.
- [96] F. Rusek, E. K. S. Au, J. B. Anderson, and W. H. Mow, "A minimum distance analysis of a certain class of two dimensional ISI channels," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, July 2008, pp. 206-210.
- [97] S. Karakulak, P. H. Siegel, and J. K. Wolf, "A parametric study of inter-track interference in bit patterned media recording," *IEEE Trans. Magn.*, vol. 46, no. 3, pp. 819-824, Mar. 2010.
- [98] 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 Int. Conf. Commun. (ICC)*, 1995, pp. 1009-1013.
- [99] A. Duel-Hallen and C. Heegard, "Delayed decision-feedback sequence estimation," *IEEE Trans. Commun.*, vol. 37, no. 5, pp. 428-436, May 1989.
- [100] G. Colavolpe, G. Ferrari, and R. Raheli, "Reduced-state BCJR-type algorithms," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 848-859, May 2001.

- [101] S. ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, no. 10, pp. 1727-1737, Oct. 2001.
- [102] D. Mackay. Encyclopedia of Sparse Graph Codes. [Online]. Available: <http://wol.ra.phy.cam.ac.uk/mackay/codes/data.html>.
- [103] A. Ashikhmin, G. Kramer, and S. Ten Brink, "Code rate and the area under extrinsic information transfer curves," in *Proc. IEEE Int. Symp. Inform. Theory (ISIT)*, 2002, p. 115.
- [104] D. Divsalar, S. Dolinar, and F. Pollara, "Iterative Turbo decoder analysis based on density evolution," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 5, pp. 891-907, May 2001.
- [105] M. C. Davey and D. MacKay, "Low-density parity-check codes over $GF(q)$," *IEEE Commun. lett.*, vol. 2, no. 6, pp. 165-167, June 1998.
- [106] D. Declercq and M. Fossorier, "Decoding algorithms for nonbinary LDPC codes over $GF(q)$," *IEEE Trans. Commun.*, vol. 55, no. 4, pp. 633-643, Apr. 2007.
- [107] R. M. Pyndiah, "Near-optimum decoding of product codes: Block turbo codes," *IEEE Trans. Commun.*, vol. 46, no. 8, pp. 1003-1010, Aug. 1998.
- [108] J. Li, K. R. Narayannan, E. Kurtas, and C. N. Georgiades, "On the performance of high-rate TPC/SPC codes and LDPC codes over partial response channels," *IEEE Trans. Commun.*, vol. 50, no. 5, pp. 723-734, May 2002.
- [109] C. Schlegel and A. Grant, *Coordinated Multiuser Communications*, Springer, 2006.
- [110] R. Motwani and P. Raghavan, *Randomized Algorithms*, Cambridge, U. K.: Cambridge Univ. Press, 1995.

Appendices

Appendix A

Proof of Proposition 6.1

Proof: Denote Q_v^k and Q_c^k as the total number of variable nodes and check nodes in a message-flow neighborhood of depth k , then from the description of the message-flow neighborhood, the values of Q_v^k and Q_c^k are dependent on k , T_c , T_s , d_v , d_c , but not on N .

Assume that $\mathbb{N}_\varepsilon^k$ is tree-like, with $k+1 < k^*$. When we branch out the neighborhood $\mathbb{N}_\varepsilon^k$ to form the neighborhood $\mathbb{N}_\varepsilon^{k+1}$, it can be shown that the probability that the newly revealed check nodes do not create any loop can be lower bounded by $(1 - Q_c^{k^*}/M)^{Q_c^{k+1}-Q_c^k}$. Similarly, the probability that the newly revealed variable nodes do not create any loop can be lower bounded by $(1 - Q_v^{k^*}/N)^{Q_v^{k+1}-Q_v^k}$. Thus, the probability that $\mathbb{N}_\varepsilon^{k^*}$ is tree-like is lower bounded by

$$(1 - Q_v^{k^*}/N)^{Q_v^{k^*}} (1 - Q_c^{k^*}/M)^{Q_c^{k^*}}. \quad (\text{A.1})$$

Note that $Md_c = Nd_v = N_e$. Therefore, for a sufficiently large value of N , it has

$$\begin{aligned}
\Pr\{\mathbb{N}_{\vec{e}}^{k^*} \text{ is not tree-like}\} &\leq 1 - (1 - Q_v^{k^*}/N)^{Q_v^{k^*}} (1 - Q_c^{k^*}/M)^{Q_c^{k^*}} \\
&\leq \frac{(Q_v^{k^*})^2 + \frac{d_c}{d_v}(Q_c^{k^*})^2}{N} \\
&= \eta/N
\end{aligned} \tag{A.2}$$

where $\eta = (Q_v^{k^*})^2 + \frac{d_c}{d_v}(Q_c^{k^*})^2$ is a constant dependent on k, T_c, T_s, d_v, d_c , but not on N . ■

Appendix B

Proof of Theorem 6.1

Proof: Denote $e_{\vec{e}_i}^k(\mathbf{c})$ as the number of erroneous messages passed along the variable-to-check edge $\vec{e}_i$, for $i = 1, \dots, N_e$, when $\mathbf{c}$ is the transmitted codeword and k rounds of the message-passing decoding algorithm have been implemented. We have

$$\begin{aligned} \mathbb{E}[e_{\vec{e}_1}^k(\mathbf{c})] &= \mathbb{E}[e_{\vec{e}_1}^k(\mathbf{c}) | \mathbb{N}_{\vec{e}_1}^k \text{ is tree-like}] \Pr\{\mathbb{N}_{\vec{e}_1}^k \text{ is tree-like}\} \\ &\quad + \mathbb{E}[e_{\vec{e}_1}^k(\mathbf{c}) | \mathbb{N}_{\vec{e}_1}^k \text{ is not tree-like}] \Pr\{\mathbb{N}_{\vec{e}_1}^k \text{ is not tree-like}\}. \end{aligned} \quad (\text{B.1})$$

From Proposition 6.1, $\Pr\{\mathbb{N}_{\vec{e}}^{k*} \text{ is not tree-like}\} \leq \eta/N$. Since the code graph is assumed to be chosen uniformly at random from the ensemble of all graphs, we have

$$\mathbb{E}[e^k(\mathbf{c})] = \sum_{i=1}^{N_e} \mathbb{E}[e_{\vec{e}_i}^k(\mathbf{c})] = N_e \mathbb{E}[e_{\vec{e}_1}^k(\mathbf{c})]. \quad (\text{B.2})$$

Thus, it can be shown that

$$N_e p^k(\mathbf{c}) - N_e p^k(\mathbf{c}) \eta/N \leq \mathbb{E}[e^k(\mathbf{c})] \leq N_e p^k(\mathbf{c}) + N_e \eta/N. \quad (\text{B.3})$$

Hence, for a sufficiently large value of N , we have

$$\Pr \left(\left| \frac{\mathbb{E}[e^k(\mathbf{c})]}{N_e} - p^k(\mathbf{c}) \right| > \epsilon \right). \quad (\text{B.4})$$

Following [23], we can construct a Doob's Martingale Process and apply Azuma's Inequality [110] to obtain

$$\Pr \left(\left| \frac{e^k(\mathbf{c})}{N_e} - \frac{\mathbb{E}[e^k(\mathbf{c})]}{N_e} \right| > \frac{\epsilon}{2} \right) \leq e^{-\mu\epsilon^2 N} \quad (\text{B.5})$$

where μ is some constant value independent of N .

Therefore,

$$\begin{aligned} \Pr \left(\left| \frac{e^k(\mathbf{c})}{N_e} - p^k(\mathbf{c}) \right| > \epsilon \right) &\leq \Pr \left(\left| \frac{e^k(\mathbf{c})}{N_e} - \frac{\mathbb{E}[e^k(\mathbf{c})]}{N_e} \right| > \frac{\epsilon}{2} \right) \\ &\quad + \Pr \left(\left| \frac{\mathbb{E}[e^k(\mathbf{c})]}{N_e} - p^k(\mathbf{c}) \right| > \frac{\epsilon}{2} \right) \\ &\leq 2e^{-\mu\epsilon^2 N}. \end{aligned} \quad (\text{B.6})$$

■

Appendix C

Proof of Theorem 6.2

Proof: From Theorem 6.1, for a sufficiently large value of N , we have $\Pr \left(\left| \frac{e^k(\mathbf{c}_i)}{N_e} - p^k(\mathbf{c}_i) \right| > \epsilon \right) \leq 2e^{-\mu'\epsilon^2 N}$, for $i = 1, 2, \dots, 2^N$. Thus,

$$\begin{aligned}
 & \Pr \left(\left| \frac{e^k(\mathbf{c}_{\text{i.u.d.}})}{N_e} - p_{\text{i.u.d.}}^k \right| > \epsilon \right) \\
 &= \sum_{i=1}^{2^N} 2^{-N} \Pr \left(\left| \frac{e^k(\mathbf{c}_i)}{N_e} - p_{\text{i.u.d.}}^k \right| > \epsilon \right) \\
 &\leq \sum_{i=1}^{2^N} 2^{-N} \Pr \left(\left| \frac{e^k(\mathbf{c}_i)}{N_e} - p^k(\mathbf{c}_i) \right| > \epsilon \right) + \sum_{i=1}^{2^N} 2^{-N} \Pr (|p^k(\mathbf{c}_i) - p_{\text{i.u.d.}}^k| > \epsilon) \\
 &\leq \sum_{i=1}^{2^N} 2^{-N} 2e^{-\mu'\epsilon^2 N} + \Pr (|p^k(\mathbf{c}_{\text{i.u.d.}}) - p_{\text{i.u.d.}}^k| > \epsilon) \\
 &= 2e^{-\mu'\epsilon^2 N} + \Pr (|p^k(\mathbf{c}_{\text{i.u.d.}}) - \mathbb{E}[p^k(\mathbf{c}_{\text{i.u.d.}})]| > \epsilon). \tag{C.1}
 \end{aligned}$$

By constructing a Doob symbol-revealing martingale sequence [62] and apply Azuma's Inequality [110], it can be proved that

$$\Pr (|p^k(\mathbf{c}_{\text{i.u.d.}}) - \mathbb{E}[p^k(\mathbf{c}_{\text{i.u.d.}})]| > \epsilon) < 2e^{-\mu''\epsilon^2 N}. \tag{C.2}$$

Hence, (6.15) follows immediately from (C.1) and (C.2). ■