MULTITRACK DETECTION FOR MAGNETIC RECORDING

A Dissertation
Presented to
The Academic Faculty

By

Shanwei Shi

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Engineering
Department of Electrical and Computer Engineering

Georgia Institute of Technology

December 2021

MULTITRACK DETECTION FOR MAGNETIC RECORDING

Thesis committee:

Dr. John R. Barry, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Gordon L. Stüber
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Matthieu R. Bloch
School of Electrical and Computer Engineering
Georgia Institute of Technology

Dr. Yao Xie
School of Industrial and Systems Engineering
Georgia Institute of Technology

Dr. Xiaoli Ma
School of Electrical and Computer Engineering
Georgia Institute of Technology

Date approved: November 23, 2021

Learning without thought is labor lost; thought without learning is perilous.

Confucius

To Ying and Hongbing

and Qing

ACKNOWLEDGMENTS

During my study in Georgia Tech, I have received a lot of help and support. First of all, I would like to express my deepest gratitude to my Ph.D. advisor Dr. John Barry. He not only taught me knowledge in books but also showed the attitude and methods to do academia. He enlightened the way ahead of me and inspired me to move forward. I cannot forget the late night we spent catching the deadline of a paper, the encouragement when I was stuck in hard problems, and the applause after we presented our work in front of sponsors.

I would like to thank Dr. Matthieu Bloch and Dr. Xiaoli Ma for providing me with constructive advice and also serving on my reading committee. I also thank Dr. Gordon Stüber and Dr. Yao Xie for serving on my dissertation defense committee. My dissertation benefits a lot from their valuable comments and suggestions.

I would also like to thank my lab mates in Communication and Information Theory Lab. Experienced senior Sarwat and Elnaz set great examples so that I can avoid detours. It is always helpful to discuss and talk with Derrick and Mohammad. Their company made the lab a warm and colorful place. Special thanks are due to the friends and colleagues at Georgia Tech, Yubing, Yan, Zoe, Le, Hao, Wei, etc., for their help in my study and life.

Finally, I would like to express my gratitude to my parents and Qing. I would never successfully finish my Ph.D. program without your love, support and tolerance.

TABLE OF CONTENTS

Acknowledgments v

List of Tables x

List of Figures xi

List of Acronyms xiv

Chapter 1: Introduction and Background 1

1.1 Introduction 1

1.2 Thesis Goal: Advanced Signal Processing for Magnetic Recording 2

1.2.1 Multitrack Detection System Design 2

1.2.2 Strategies for Choosing Parameters 3

1.2.3 Neural Networks for Magnetic Recording 3

1.3 Background and Literature Survey 4

1.3.1 Magnetic Recording System 4

1.3.2 Pattern-Dependent Noise Prediction 7

1.3.3 Two-Dimensional Magnetic Recording 9

1.3.4 Minimum-BER Based Detection 12

1.3.5 Iterative (Turbo) Decoding 15

1.3.6 Machine Learning for Magnetic Recording 16

Chapter 2: Multitrack Detection with Pattern Dependent Noise Prediction . . . 18

2.1 Introduction 18

2.2 Joint MMSE Optimization of Equalizer and Target 18

2.3 Multitrack Detection with Multitrack PDNP 21

2.3.1 A Multitrack Pattern-Dependent and Autoregressive Model for Noise 21

2.3.2 The Joint ML Solution to the Multitrack Detection Problem 22

2.4 Estimation of PDNP Parameters 25

2.4.1 Non-Pattern-Dependent Multitrack Noise Prediction 26

2.4.2 Estimating AR Channel Parameters via Training 27

2.4.3 Self Training Scheme for PDNP 29

2.5 Quantitative Results 30

2.5.1 Ehime Waveforms 30

2.5.2 PDNP Parameters via Training 31

2.5.3 PDNP Parameters via Self Training 34

2.6 Summary 35

Chapter 3: Minimum-Bit-Error Rate Tuning 38

3.1 Introduction 38

3.2 The Cost Function and AMBER Algorithm 39

3.2.1 AMBER for Viterbi without PDNP 42

3.2.2 AMBER for 1D-PDNP 43

3.2.3 AMBER for 2D-PDNP 45

3.3 The Exponential Assumption and Hyper-Parameter Optimization 46
3.3.1 Exponential Assumption 46
3.3.2 Threshold 48
3.3.3 Hamming-Weighted Step Size 50
3.4 Quantitative Results 51
3.4.1 Performance of AMBER PDNP Detection 51
3.4.2 Performance of AMBER 2D-PDNP Detection 53
3.4.3 Optimize the Parameter Space 55
3.5 Summary 58

Chapter 4: Minimum-Frame-Error Tuning for Turbo Detection 60

4.1 Introduction 60
4.2 The EXIT Chart 60
4.3 Adaptive Minimum-Frame-Error Rate Algorithm 64
4.3.1 AMFER for Single Track Detection 66
4.3.2 AMFER for Multitrack Detection 67
4.4 Quantitative Results 69
4.4.1 Linear ISI Example 69
4.4.2 Single-Track Detection on Ehime waveforms 69
4.4.3 Multitrack Detection on Ehime waveforms 73
4.5 Summary 77

Chapter 5: Neural Network for Magnetic Recording 79

5.1 Introduction 79

5.2 Classic Neural Network Structures 79

5.3 Neural Network for Hard-Output Detection 85

5.4 Summary 90

Chapter 6: Conclusions and Future Work 91

6.1 Contributions 91

6.2 Future Work 93

Appendices 96

Appendix A: Convergence Condition and Proof of Algorithm 3 97

Appendix B: Derivation of (4.1) 98

References 99

Vita 106

LIST OF TABLES

3.1 BER and number of parameters adapted for all parameter schemes. 56

5.1 Output size and number of parameters of each layer in the RNN shown in Fig. 5.5. 87

LIST OF FIGURES

1.1 (Left) Annual size of the global datasphere [2]. (Right) Where data will be stored in the datasphere [2]. 2

1.2 (Left) Inside look of a typical HDD [9]. (Right)The analogy of the magnetic read channel to a communication system. 5

1.3 A typical read channel for a magnetic recording system. 6

1.4 Readback waveforms (red) simulation based on Voronoi grains. The average grains/bit decreases from top to bottom. When there are only a few grains per bit, the irregular boundaries corrupt the waveform as shown in the bottom figure. 8

1.5 A synchronous written multitrack detection system (M readers detect N tracks). 11

1.6 Block diagram of a channel, equalizer, and a memoryless decision device [38]. 12

1.7 A serial-concatenated turbo encoder (a) consists of an outer encoder, an interleaver and an inner code. A turbo decoder (b) consists of two APP decoders. 15

2.1 A multitrack detection example, where two readers ($N = 2$) are used to detect two tracks ($M = 2$). 19

2.2 The signal and 2D media noise model for the special case of $N_p = 4$, $I = 1$, and $J = 3$. 23

2.3 The self training scheme. 30

2.4 Geometry of the Ehime waveform model, showing the five-shingled tracks and 25 possible reader locations [30]. 31

2.5 BER performance of multitrack Viterbi, multitrack PDNP (4-bit) and multitrack PDNP (6-bit). 32

2.6 The bathtub curves with multitrack Viterbi and multitrack PDNP (6-bit). . . 33

2.7 BER versus threshold at track pitch 24.1 nm in the second model estimation. 35

2.8 BER evolution of self training algorithm at track pitch 26.1 nm and threshold 6. 36

3.1 Illustration of the parameters to be optimized, for two scenarios: (a) for single-track detection using a single reader; (b) for multitrack detection using multiple readers. 40

3.2 Margin PDF and its tail fit to an exponential function. 47

3.3 The fixed-point relationship between τ and $1/b(\mathbf{\Theta}(\tau))$. The simulation is run in training mode. 50

3.4 BER contour with trajectories of AMBER parameters w/o Hamming-weight factor. 52

3.5 Learning and testing curves for the AMBER PDNP algorithm. 53

3.6 Parameter evolution for AMBER PDNP. 54

3.7 Learning and testing curves for the AMBER 2D-PDNP algorithm. 55

3.8 BER performance of all schemes. 57

3.9 Performance and complexity trade-off of different schemes. 58

4.1 A soft-output detector interacting with an LDPC decoder in an iterative (turbo) fashion. 61

4.2 EXIT charts of Example 1 under different SNR's. 64

4.3 EXIT chart with MMSE and AMFER parameters. 70

4.4 FER vs SNR with MMSE and AMFER parameters. 71

4.5 EXIT charts based on different detectors. 72

4.6 Measured FER performance of MMSE and AMFER parameters w/o PDNP. 73

4.7 EXIT chats with MMSE parameters (yellow), AMFER parameters trained by $I_2^* = 0.2$ (green) and AMFER parameters trained by $I_2^* \in [0, 0.5]$. 74

4.8 (a) Average BER measured by the sign of $\boldsymbol{\lambda}_1$ as turbo iteration increases. (b) Evolution of measured FER as turbo iteration increases. 75

4.9 Measured FER vs electronic SNR with MMSE and AMFER parameters. . . 76

4.10 Measured FER vs track pitch with MMSE and AMFER parameters. 77

5.1 A typical structure of the artificial neural network: one input layer (labeled in blue), one or more hidden layers (labeled in grey) and one output layer (labeled in red). Circles represent neurons and arrows represent the direction of data flow. 80

5.2 The standard recurrent neural network. 83

5.3 Data flow inside a long short-term memory cell. 84

5.4 (a) A conventional detection system. (b) A neural network detection system. 85

5.5 The recurrent neural network aims to do sequence detection. The network consists of a convolution layer, a bidirectional LSTM layer and a time-distributed dense layer. 86

5.6 The BER evolution of training and validation data as the epoch increases. . 88

5.7 BER performance of the RNN and conventional structures (with MMSE and AMBER parameters). 89

6.1 An iterative approach to jointly optimizing an LDPC code and a soft-output detector. 94

LIST OF ACRONYMS

1D-PDNP single-track pattern-dependent noise prediction

2D-PDNP multitrack pattern-dependent noise prediction

ADC analog to digital converter

AMBER adaptive minimum-bit-error rate

AMFER adaptive minimum-frame-error rate

APP *a posteriori*

AR autoregressive

BCJR Bahl-Cocke-Jelinek-Raviv algorithm

BER bit-error rate

CNN convolutional neural network

DNN deep neural network

FER frame-error rate

GPR generalized partial response

HDD hard disk drive

ISI intersymbol interference

LLR log-likelihood ratio

LMS least mean squares

LSTM long short-term memory

MBER minimum-bit-error rate

MIMO multiple-input and multiple-output

ML maximum likelihood

MMSE minimum-mean-square error

MSE mean-square error

PDF probability density function

PDNP pattern-dependent noise prediction

PMR perpendicular magnetic recording

PRML partial response maximum likelihood

ReLU rectified linear unit

RNN recurrent neural network

SMR shingled magnetic recording

SNR signal-to-noise ratio

SOVA soft-output Viterbi algorithm

TDMR two-dimensional magnetic recording

SUMMARY

The exploding demand for cloud storage is motivating a push for higher areal densities, with narrower track pitches and shorter bit lengths. The resulting increase in interference and media noise requires improvements in read channel signal processing to keep pace. This dissertation develops advanced signal processing algorithms for magnetic recording systems to increase the areal density.

The advent of multiple readers in magnetic recording opens the door to multitrack detection, in which multiple tracks are detected jointly. Multitrack detection is a key enabler for both coding across tracks and crosstrack noise prediction, neither of which can be fully exploited using single-track detectors. A dominant impediment in magnetic recording is pattern-dependent media noise, and its impact will only grow more severe as areal densities increase. A widely used strategy in single-track detection for mitigating media noise in a trellis-based detector is pattern-dependent noise prediction. This thesis proposes the multitrack pattern-dependent noise-prediction algorithm as a solution to the joint maximum-likelihood multitrack detection problem in the face of pattern-dependent autoregressive Gaussian noise.

The magnetic recording read channel has numerous parameters that must be carefully tuned for best performance; these include not only the equalizer coefficients but also any parameters inside the detector, some of which may be pattern dependent, including signal levels, predictor coefficients, and residual noise variances. Conventional tuning strategies based on a minimum-mean-squared error criterion are not optimal in terms of bit-error rate and frame-error rate. Because the number of states grows exponentially with the number of tracks being detected, a multitrack detector has far more parameters than a single-track detector. This thesis proposes two alternative tuning strategies (1) to minimize the bit-error rate after detection, and (2) to minimize frame-error rate after error-control decoding. We also propose a method to reduce redundant parameters.

Furthermore, this thesis proposes and designs a neural network read channel architecture, and compares the performance and complexity with these traditional signal processing techniques.

CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Introduction

We are living in an increasingly digital world, where the amount of digital data that is generated and stored continues to grow. For example, more than 500 hours of video are uploaded to YouTube every minute [1], which requires approximately 1.5 TB of space (assuming 1080p at 60 FPS). As illustrated in Fig. 1.1 (Left), the amount of customer generated data will grow from 33 Zettabytes (1 ZB = 10^9 TB) in 2018 to 175 ZB by 2025 [2].

Users increasingly store their personal data on cloud storage services such as Dropbox, Google Drive, and iCloud, which inspires the growth of the storage industry. The International Data Corporation predicts that more bits will be stored on the public cloud than on consumer devices, as shown in Fig. 1.1 (Right). Furthermore, cloud storage generally replicates each user bit and stores multiple copies, as a form of redundancy against storage errors [3].

Solid-state drives have overtaken hard disk drives as the storage technology of choice for many consumer devices, largely because solid-state drives have faster read and write speeds, and lower energy consumption. Nevertheless, hard disk drives dominate in data centers. This is because a typical hard disk drive (HDD) has a cheaper cost per bit, larger capacity, and longer endurance.

Areal density is a measure of how many bits can be reliably stored per unit area of the storage medium. It is a vital metric for HDD's because it generally dictates the cost required to store each bit; higher areal density means lower cost. The high demand for data storage inspires new technologies for HDD's. New writing and reading methods,

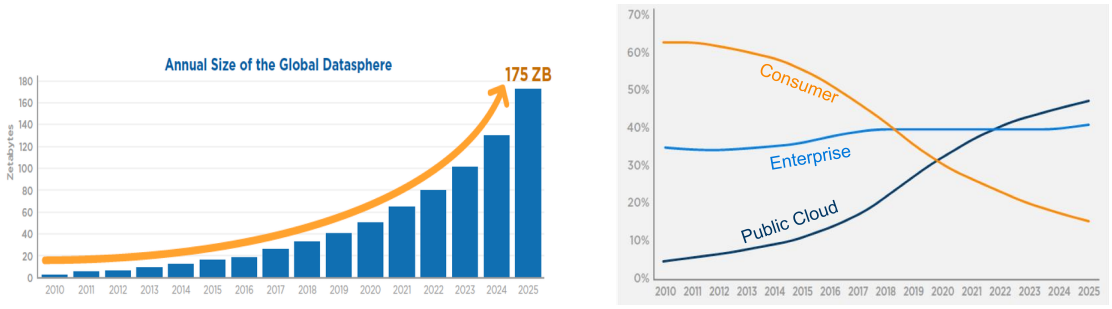


Figure 1.1: (Left) Annual size of the global datasphere [2]. (Right) Where data will be stored in the datasphere [2].

medium materials, and signal processing techniques are continuously being developed by researchers and industry so as to increase the areal density of HDD's.

A conventional HDD uses a single reader (sensor) to recover the stored bits, but an industry roadmap for HDD technology predicts that multiple readers will soon be the standard [4]. Hard drives with two readers are just now emerging in the marketplace. However, none of these products use the technique of *multitrack detection*, which employs multiple readers to jointly detect and recover multiple tracks. The concept of multitrack detection was first proposed in 1993 [5], and has recently become a promising candidate to break through the current limit of areal density [6].

1.2 Thesis Goal: Advanced Signal Processing for Magnetic Recording

The goal of this thesis is to devise advanced signal processing algorithms for magnetic recording to reach higher areal density. The thesis will deliver an architecture of multitrack detection for a magnetic recording system with two or more read heads, two strategies for tuning related parameters and a neural network for hard-output detection.

1.2.1 Multitrack Detection System Design

One target of this research is to design a multitrack detection structure. The motivation for this research is the observation that multitrack detection not only doubles the throughput,

but also enables the exploitation of crosstrack modulation coding, crosstrack error-control coding, and multitrack noise prediction. The various components of single-track detection including the equalizer, target design, pattern-dependent noise prediction, coding, are well understood and mature, but their multitrack counterparts are still being uncovered. We will jointly design an equalizer and target to deal with multiple tracks of input waveforms and mitigate intertrack interference and intersymbol inference. We will model 2D media noise and propose a multitrack sequence detector with multitrack pattern-dependent noise prediction. We will also propose a strategy to implement crosstrack error-control coding so that the detector can work well with a decoder.

1.2.2 Strategies for Choosing Parameters

The conventional parameter choice is based on a minimum-mean-square error criterion. However, prior work has seldom tied the metrics to the performance that are of interest to the end user, such as bit-error rate and frame-error rate. Further, including the equalizer and detector, the number of parameters will typically be in the tens or hundreds so that it is not practical to optimize via brute-force search. We propose new strategies for tuning these parameters based on different criteria. We first propose a tuning strategy aiming at minimizing the bit-error rate after detection. We then propose a tuning strategy aiming to minimize the frame-error rate after the error-control decoder when that decoder works iteratively with the detector in a turbo fashion. We will optimize the parameter space to avoid redundant or negligible parameters.

1.2.3 Neural Networks for Magnetic Recording

Neural networks have become a popular solution for complex problems. We design and train a neural network to minimize the bit-error rate for magnetic recording and compare its performance and complexity with traditional detectors.

This thesis is organized as follows. In the rest of this chapter, we present the

background and related work about multitrack detection for magnetic recording. In Chapter 2 we provide a multitrack detection framework and its minimum-mean-square error solution to parameters. In Chapter 3 we propose a strategy to tune parameters aiming at minimizing bit-error rate. In Chapter 4 we propose a minimum-frame-error rate tuning strategy when detection is followed by decoding. In Chapter 5, we develop neural network techniques for magnetic recording. Finally in Chapter 6, our conclusions and future work are presented.

1.3 Background and Literature Survey

1.3.1 Magnetic Recording System

A typical HDD consists of a number of platters stacked vertically with one read/write head mounted on the tip of an actuator arm for each platter, see Fig. 1.2. As the platters spin, the drive heads can be moved to reach any point of each platter. Each platter has thousands of thin concentric bands called tracks. Information bits are read and written by sectors on a track. The size of a sector is fixed for a typical HDD, e.g., 512 bytes for traditional HDD's and 4K bytes for new HDD's. The firmware of a typical HDD is in charge of controlling actuator arms and signal processing of read and write operations.

A conventional HDD uses a single reader (sensor) to receive magnetic signals. Hard drives with two readers are just now emerging in the marketplace. For example, Western Digital launched a product called the *HGST Ultrastar DC HC530* in 2018, which uses two readers to detect and recover a single track with a single-track detector [7]. Seagate Technology will launch a new product called the *MACH.2* soon that has two reading arms working independently [8].

The recording method is also updating rapidly in the HDD industry. The traditional longitudinal recording has been replaced by perpendicular magnetic recording (PMR) and shingled magnetic recording (SMR). The perpendicular recording head produces magnetic flux perpendicular to the recording layer. A small area of grains on the media is polarized to

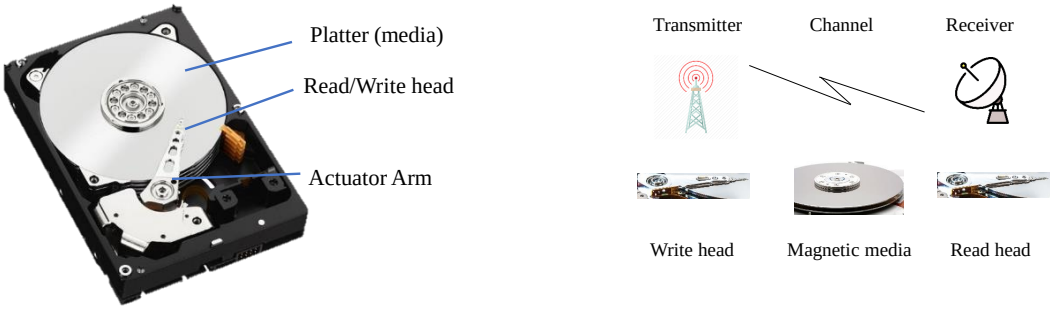


Figure 1.2: (Left) Inside look of a typical HDD [9]. (Right)The analogy of the magnetic read channel to a communication system.

either negative or positive to represent binary bits. PMR is reported to have more than three times the areal density of longitudinal recording [10]. Instead of writing data parallelly on non-overlapping tracks like PMR, SMR records a new track overlapping part of the previously written track, which allows for higher areal density [11].

Energy-assisted magnetic recording is considered as the next-generation technique for HDD's. As the areal density increases, drives need higher energy barriers in media to maintain thermal stability [12]. Meanwhile, actuator heads must generate stronger magnetic fields against energy barriers. This is where energy-assisted magnetic recording technology comes in. With the help of heat (heat-assisted magnetic recording, HAMR) or microwaves (microwave-assisted magnetic recording, MAMR), heads can easily change the polar magnetization of media. It is still unclear which assistant method comes out on top.

A magnetic recording system writes information bits on a disk and reads information from the disk. It functions as a communication system so that it is modeled as a communication system: transmitter, channel, and receiver as shown in Fig. 1.2. The transmitter consists of an error-control encoder, a modulation encoder, a modulator and a write head. The write head performs magnetization by hovering near the surface of a magnetic medium. The magnetic medium can be considered a communication channel contaminated by both electronic and medium noise. A read head, a read channel, a

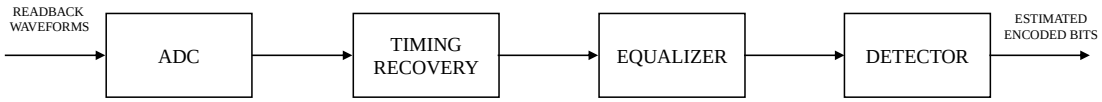


Figure 1.3: A typical read channel for a magnetic recording system.

modulation decoder and an error-control decoder compose the receiver [13][14]. The read head detects the current magnetization and achieves readback waveforms. The so-called *read channel* in magnetic recording usually includes an analog to digital converter (ADC), a timing recovery block, an equalizer, and a symbol detector, as shown in Fig. 1.3. This thesis focuses on the read channel.

The timing recovery block corrects and compensates for the misalignment between the ADC sampling clock and the actual arrival time of the bits. Conventional analog timing recovery is replaced by a fully digital timing recovery scheme known as interpolated timing recovery [15]. The sampled signal after ADC is asynchronous but gets synchronized using interpolation techniques based on the timing information achieved from a digital phase-locked loop.

As the areal density increased, the conventional peak detector was replaced by the partial response maximum likelihood (PRML) sequence detector [16] in the 1990s due to the heavy intersymbol interference (ISI). ISI arises because the tail of one pulse does not die away before the next pulse is transmitted. Rather than eliminating or canceling the ISI like the peak detector, the read channel artificially introduces ISI and the equalizer and the PRML detector handles it afterward, which enables higher density and paves the way for other advanced detection and coding techniques. A partial response is a short small integer-valued channel response $\mathbf{g} = [1, g_1, ..., g_\mu]^T$ filtering the transmitted bits. Typical classes of the PR are PR4 ($\mathbf{g} = [1, 0, -1]^T$) and EPR4 ($\mathbf{g} = [1, -1, 1, -1]^T$). Then the equalizer coefficient vector $\mathbf{c}$ is chosen to minimize the mean-square error (MSE) between the equalizer output and the target signal, which is achieved by convolving bits and PR

coefficients, i.e.

$$\text{MSE} = E((\mathbf{c}^T\mathbf{r}_k - \mathbf{g}^T\mathbf{a}_{k-d})^2), \tag{1.1}$$

where $r_k = [r_k, r_{k-1}, ..., r_{k-N_c+1}]^T$ is the vector of relevant waveform samples and $\mathbf{a}_{k-d} = [a_{k-d}, ..., a_{k-d-\mu}]^T$ is the vector of written bits with some delay d. Therefore, the PR coefficients are also called the target. After equalization, the signal is detected by a sequence detector with the target. The generalized partial response (GPR) extends the integer-valued target to arbitrary coefficients [17], which yields better performance. The optimization needs a constraint on the target to avoid a trivial solution.

1.3.2 Pattern-Dependent Noise Prediction

Like most communication channels, the magnetic recording channel is corrupted by thermal noise. The magnetic recording system also needs to combat a unique impediment: media noise [13]. Media noise is generated because of the randomly sized and positioned grains on the magnetic media. As there are fewer grains per bit, the boundaries of bit transitions become random and irregular, as shown in Fig. 1.4. This effect leads to inaccurate readback waveforms. Consequently, the key feature of media noise is that it depends on the data being written, and in particular is more pronounced in the vicinity of bit transitions. A bit segment with more transitions causes severer media noise.

The noise after the PR equalizer is not white. A Viterbi detector [18] that treats the noise as white will perform poorly. One method to whiten this noise is to integrate noise prediction into the sequence detector [19][20]. However, additive Gaussian white noise only accounts for part of the total noise since media noise can be dominant especially for high areal density disks. To handle the media noise, a pattern-dependent noise whitening procedure is commonly used, which gives rise to the pattern-dependent noise prediction (PDNP) detector proposed in [21]. The authors assume that the read channel is contaminated with ISI and autoregressive Gauss-Markov noise. At time k the noise n_k

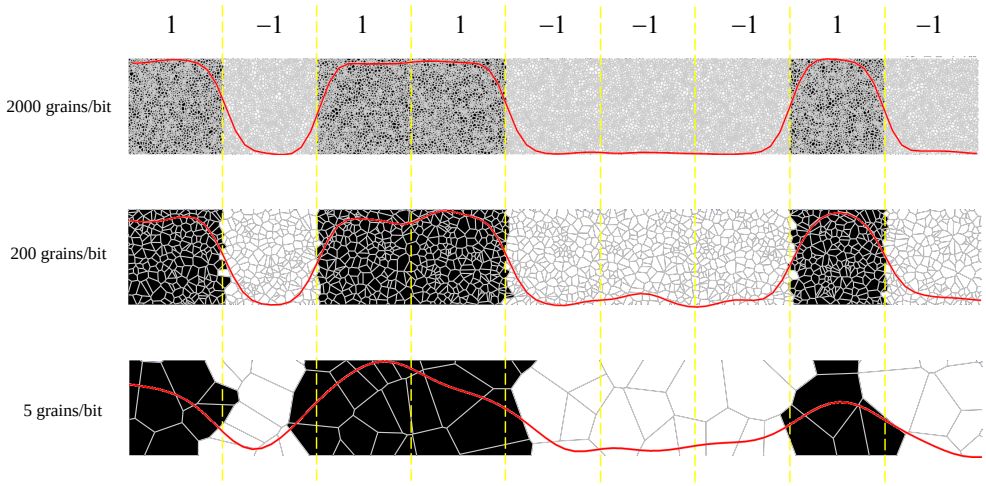


Figure 1.4: Readback waveforms (red) simulation based on Voronoi grains. The average grains/bit decreases from top to bottom. When there are only a few grains per bit, the irregular boundaries corrupt the waveform as shown in the bottom figure.

with Markov memory length L is dependent on its bit pattern $\underline{a}_k$, i.e.,

$$n_k = \underline{b}(\underline{a}_k)^T \underline{n}_{k-1}^{k-L} + \sigma(\underline{a}_k) w_k, \tag{1.2}$$

where $\underline{b}(\underline{a}_k)$ is the pattern-dependent autoregressive filter on the previous noise vector $\underline{n}_{k-1}^{k-L}$, w_k is a standard normal noise process, and $\sigma(\underline{a}_k)$ is the pattern-dependent standard deviation. Based on this model, the Gaussian maximum likelihood (ML) branch metric from state x_{k-1} to x_k, given observed vector $\underline{z}_k^{k-L}$, can be expressed as

$$\mathcal{M}_{ML}(\underline{z}_k^{k-L}, x_{k-1}, x_k) = \ln \sigma^2(\underline{a}_k) + \frac{([-\underline{b}(\underline{a}_k)^T \; 1](\underline{z}_k^{k-L} - \underline{Y}(x_{k-1}, x_k)))^2}{\sigma^2(\underline{a}_k)}, \tag{1.3}$$

where $\underline{Y}(x_{k-1}, x_k)$ is the vector of the ideal channel outputs.

[22] explores the PDNP from a linear prediction viewpoint without any assumption on the noise model. The noise predictor whitens the noise and reduces its variance for each bit pattern. Although it arrives at the same branch metric as in [21], the derivation is intuitive. Later, the PDNP Viterbi algorithm is extended to the Bahl-Cocke-Jelinek-Raviv algorithm

(BCJR) and the soft-output Viterbi algorithm (SOVA) [13].

Pattern-dependent noise prediction is one of the most important signal processing techniques for magnetic recording in the last 20 years. This technique is widely recognized and used by the HDD industry and extended into many other fields beyond data storage, such as signal and image processing, and data science [23].

1.3.3 Two-Dimensional Magnetic Recording

The conventional single-track magnetic recording is predicted to reach its limit at around 1 $\mathrm{Tbit/in^2}$ due to the thermal stability [24]. The push for a higher areal density and the advent of a shingled writing technique motivate the use of two-dimensional magnetic recording (TDMR). TDMR does not require new media or new head technologies, but only requires new signal processing and coding methods. The ideal TDMR handles the 2D readback signal in both crosstrack and downtrack directions, like an image. While the TDMR has a promising future, it is still a long way from productization. Fortunately, multitrack magnetic recording, which reads and detects several tracks (much fewer compared to the length of a track), becomes a stepping-stone for TDMR. In the HDD industry multitrack magnetic recording is sometimes also called TDMR. We will introduce TDMR detection and multitrack detection separately due to the different models used.

2D Page Detection

An ideal TDMR system organizes the data to be written in a 2D array whose two dimensions are comparable in size and interchangeable. In the reading process, sensing the magnetization is modeled as convolving with a read-head response [25]. The 2D readback process is modeled as a matrix channel response capturing all ISI and intertrack interference (ITI), and then convolved with a matrix of recorded bits.

GPR equalizer and target design are studied in [26] and [27]. The 2D detector and its

complexity are discussed in [27] and [28]. Data-dependent noise prediction (DDNP) for TDMR is proposed in [29], where the noise is predicted using the estimated noise nearby the current bit. The DDNP detection algorithm is observed to give over 10% gains in areal density over a 2D soft-output Viterbi algorithm without noise prediction when no grain flips are considered.

Multitrack Detection

A published roadmap predicted that, in order to realize the benefits of TDMR, multitrack detection will be required within the next five to ten years [4]. Multitrack detection is fed multiple readback waveforms and similarly detects multiple tracks. A synchronized written multitrack detection system is shown in Fig. 1.5. Shingled writing enables narrow track pitches, resulting in significant intertrack interference as well as a nonlinear transition shift in both downtrack and crosstrack directions. Multitrack detection opens the door to exploiting error-control coding and modulation coding across tracks, as well as crosstrack pattern-dependent noise prediction.

The move from single-track to multitrack detection requires a substantial redesign of many system components, including the geometry of the readers, the equalizers, the target, and the detection algorithm. The equalizer that takes in multiple streams of sampled readback waveforms and shapes the signal to fit a target is known as a 2D equalizer. In [30], the authors use a 2D equalizer and a 1D target to investigate the optimization of bit geometry and two-reader geometry when detecting a single track. The performance gain from one reader to two readers is clearly shown.

2D patterns are used for noise prediction in [31], but the tracks are detected independently with 1D noise predictors. In [32], decisions from a previously detected track are used to detect future tracks based on 2D patterns and single-track pattern-dependent noise prediction (1D-PDNP). In [33], the authors extend this work to include 2D write precompensation based on three-track patterns.

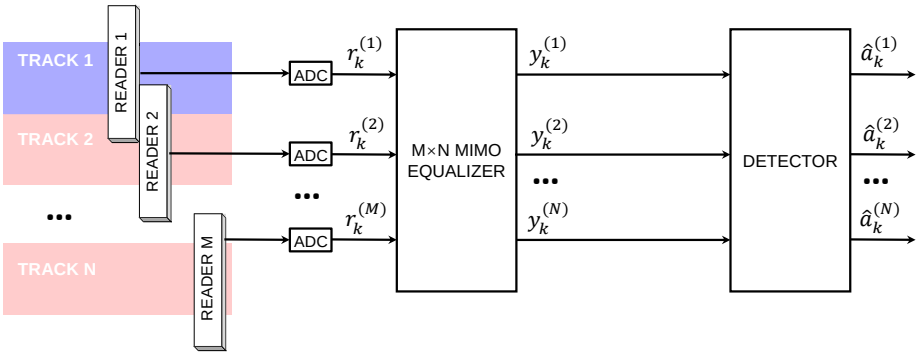


Figure 1.5: A synchronous written multitrack detection system (M readers detect N tracks).

In [34], [35], and [36], the authors propose an alternative way to implement maximum-likelihood detection for multihead multitrack systems called the weighted sum subtract joint detector. This detector uses a different trellis, which is independent of ITI. Therefore it is efficient to adapt for the time-varying ITI environment.

Synchronization for Multiple Tracks

The conventional way to recover data from readback waveforms of multiple asynchronous tracks is to synchronize and detect each track separately, which goes back to the single-track situation. However, ITI should be mitigated as much as possible before detection. An alternative way is to jointly synchronize and detect multiple tracks simultaneously. [37] proposes an innovative rotating-target (ROTAR) algorithm to handle the joint detection of multiple asynchronous tracks. Synchronization for multitrack detection is no longer performed as a separate block in single-track detection. This algorithm jointly performs the synchronization and detection using Viterbi detection with a time-varying target. The authors show that the ROTAR algorithm outperforms the conventional way by 1 dB when achieving the BER of 10^{-4}. The time-varying target matched equalizer for an unknown channel is also proposed in [14].

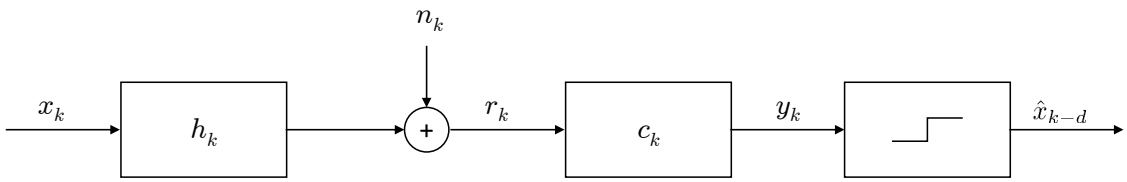


Figure 1.6: Block diagram of a channel, equalizer, and a memoryless decision device [38].

1.3.4 Minimum-BER Based Detection

The equalizers, targets, and predictors discussed above are designed to minimize the mean-square error since it is a quadratic function that has a closed-form solution for its single minimum. Because of this advantage, the minimum-mean-square error (MMSE) criteria is widely used. However, the criteria for a hard-decision detector is the bit-error rate (BER) that has no direct relationship with the MSE. It means an equalizer that minimizes the MSE does not necessarily minimize the BER. As a result, how to choose these parameters based on minimum-bit-error rate (MBER) criteria has drawn researchers' attention. The MBER algorithms are studied for different channels as follows.

Full-Response Channel

In [38], the linear equalizer based on the MBER criterion is studied for a typical linear ISI channel with Gaussian noise and a memoryless detector. The system diagram is shown in Fig. 1.6. Binary input x_k at time k drawn from $\{\pm 1\}$ passes through a channel with impulse response h_k. r_k, the sum of the channel output and Gaussian noise n_k with variance σ^2, is the input of a linear equalizer with the coefficient c_k at time k. The sign detector makes the decision $\hat{x}_{k-d}$ on the equalized signal y_k at time k, where d accounts for the delay in the channel and the equalizer. The authors define the signal vector as $\mathbf{s}_i = \mathbf{H}\tilde{\mathbf{x}}_i$, where $\mathbf{H}$ is the Toeplitz convolution matrix of h_k, and $\tilde{\mathbf{x}}_i$ is the i-th possible input vector among all L vectors. The authors show that when a channel is known and equalizable, the BER is an

average of a set of Q functions:

$$\mathrm{BER} = \frac{1}{L}\sum_{i=1}^{L} Q(\frac{\mathbf{c}^T\mathbf{s}_i}{||\mathbf{c}||\sigma}). \tag{1.4}$$

There is no unique and closed-form solution for the exact minimum-BER equalizer coefficients denoted by $\mathbf{c}_{\mathrm{EMBER}}$. However, they achieve a lemma that an equalizer $\mathbf{c}$ that minimizes the BER of an equalizable channel must satisfy

$$\mathbf{c} = a\frac{1}{L}\sum_{i=1}^{L} e^{\frac{1}{2}(\frac{\mathbf{c}^T\mathbf{s}_i}{||\mathbf{c}||\sigma})^2} = af(\mathbf{c}), \tag{1.5}$$

where $a > 0$ and $f(\mathbf{c})$ can be considered a linear combination of the signal vector $\mathbf{s}_i$. This lemma demonstrates that $\mathbf{c}_{\mathrm{EMBER}}$ is proportional to the linear combination of all the signal vectors. If one signal vector's weight dominates among all the signal vectors, $\mathbf{c}_{\mathrm{EMBER}}$ will be approximately proportional to that signal vector. The deterministic EMBER algorithm is derived as

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \lambda f(\mathbf{c}_k), \tag{1.6}$$

where λ is the step size. For an adaptive equalization with an unknown channel, the authors provide the stochastic adaptive minimum-bit-error rate (AMBER) algorithm:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \lambda \mathbb{1}_{x_{k-D}y_k<0} x_{k-D}\mathbf{r}_k. \tag{1.7}$$

This algorithm has an expression and complexity similar to the least mean squares (LMS) algorithm and only has an extra indicator function than the sign-LMS algorithm. The AMBER algorithm only updates the equalizer when an error is made. The numeric results demonstrate that the EMBER equalizer can outperform the MMSE equalizer by 6.5 dB with 3-tap at the BER of 10^{-5}. The gap shrinks as the number of taps increases. Since the approximation $Q(z) < \frac{1}{z\sqrt{2\pi}}e^{-\frac{z^2}{2}}$ for $z > 0$ is used in the derivation, the AMBER algorithm no longer minimizes the BER exactly. However, the simulation results show

that the AMBER algorithm converges to a place closely near the minimum. The MBER decision feedback equalizer is also explored in the author's thesis [39].

Partial-Response Channel

The BER of ML sequence detection has been studied since it was invented [40] [41]. However, there is no exact expression for the BER. It has been only approximated by the probability of the minimum distance error event at a high signal-to-noise ratio (SNR).

[42] studies the near MBER equalizer for partial response ML sequence detection. The authors express the sequenced amplitude margin of bit-error sequence e as

$$S(e) = 4(\underline{\delta}_e^T \underline{\delta}_e - X_e), \tag{1.8}$$

where $\underline{\delta}_e$ is the convolution of target g and bit-error sequence e, X_e is the correlation between error signal and $\underline{\delta}_e$. Given the bit sequence b and bit-error sequence e, the probability that this error event happens is

$$P(\text{error event}|b, e) = Pr(\underline{\delta}_e^T \underline{\delta}_e < X_e). \tag{1.9}$$

The authors assume that the probability distribution of $X_e > \underline{\delta}_e^T \underline{\delta}_e$ is Gaussian with mean μ_e and variance σ_e^2 so that (1.9) results in a Q function. Meanwhile, $E(\mathbb{1}_{X_e > \underline{\delta}_e^T \underline{\delta}_e} X_e)$ approximates in proportion to the same Q function, i.e.

$$P(\text{error event}|b, e) = Q(\frac{\underline{\delta}_e^T \underline{\delta}_e - \mu_e}{\sigma_e}) \approx \frac{1}{\underline{\delta}_e^T \underline{\delta}_e} E(\mathbb{1}_{X_e > \underline{\delta}_e^T \underline{\delta}_e} X_e) \tag{1.10}$$

It means minimizing $E(\mathbb{1}_{X_e > \underline{\delta}_e^T \underline{\delta}_e} X_e)$ is approximate to minimize the (1.9). It is hard to directly minimize $P(\underline{\delta}_e^T \underline{\delta}_e < X_e)$ (i.e. $E(\mathbb{1}_{X_e > \underline{\delta}_e^T \underline{\delta}_e})$) because it is non-differentiable. However, the stochastic gradient descent algorithm to minimize $E(\mathbb{1}_{X_e > \underline{\delta}_e^T \underline{\delta}_e} X_e)$ is easy to derive.

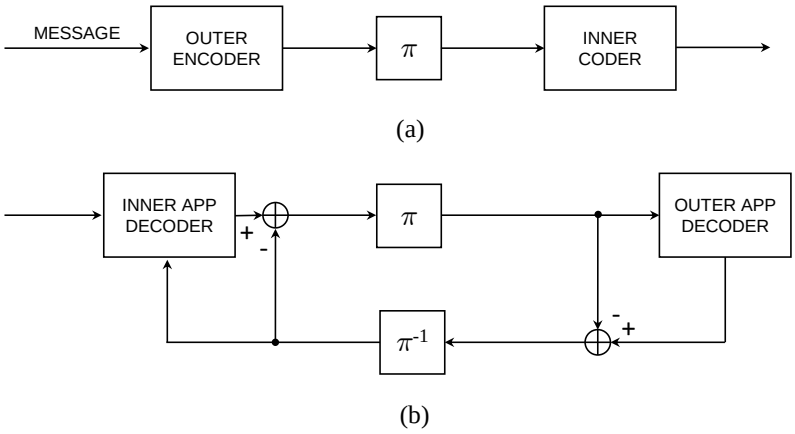


Figure 1.7: A serial-concatenated turbo encoder (a) consists of an outer encoder, an interleaver and an inner code. A turbo decoder (b) consists of two APP decoders.

Based on the algorithm to approximately minimize the probability of error event, the stochastic gradient search algorithm of an empirical BER is approximated by scaling the step size with the hamming weight of the bit-error sequence. The simulation results show that in an idealized optical storage channel, the proposed algorithm outperforms the LMS by 3.4 dB and 1.2 dB with 3-tap and 5-tap targets at BER of 10^{-4} and 10^{-5}, respectively.

Furthermore, the AMBER target is explored in the author's thesis [43] and the authors find the performance of the AMBER target is very close to that of the monic constraint MMSE target.

1.3.5 Iterative (Turbo) Decoding

Turbo codes construct a powerful error-control code with two or more easily decodable codes concatenated in parallel or serial. The receiver consists of two or more decoders respectively. They share information with each other and form a joint decoder. Turbo codes have two structures: parallel-concatenated codes (PCC) [44] and series-concatenated codes (SCC) [45]. We will introduce SCC in detail since it is widely used in magnetic recording systems.

A generic serial-concatenated encoder/decoder is shown in Fig. 1.7. The encoder

consists of an outer encoder and an inner encoder separated by a pseudorandom interleaver. The decoding is a reverse procedure of the encoding. The pair of *a posteriori* (APP) decoders exchange information iteratively. The outer decoder has no access to the channel observations but a priori information from the inner decoder. The inner code must be recursive for best performance but the outer code need not be [41].

Techniques for analyzing the convergence of iterative decoding schemes have been widely studied. Density evolution for low-density parity-check (LDPC) codes was developed by tracking the probability distributions of extrinsic log-likelihood ratios (LLRs) [46]. This tracking is simplified considerably when the probability density function (PDF) depends on only a single parameter. Other single-parameter metrics include the fidelity of [47] and the mutual information of [48]. In [48], Ten Brink developed the *extrinsic information transfer (EXIT)* chart to visualize the evolution of LLR's and analyze the convergence behavior. Prior work regarding EXIT chart has been limited to the design of error-control codes [49, 50, 51], *not detectors*.

The turbo coding for magnetic recording was studied in [52, 53]. Since the channel is a PR channel, it naturally leads to a rate-one inner code, similar to a recursive convolutional code. The inner APP decoder is designed for the precoded PR channel. Any high rate, block or convolutional code can be a good candidate for the outer code.

1.3.6 Machine Learning for Magnetic Recording

The rise of machine learning has changed all walks of life. Several papers apply machine learning techniques to magnetic recording channels.

[54] proposes a convolutional neural network (CNN) with three hidden layers. Waveforms from two readers hanging on two adjacent tracks are the input and the goal is to detect one track. Simulation results show that CNN can mitigate ITI from the adjacent track. [55] proposes a deep neural network (DNN) based media noise predictor cooperating with a BCJR detector. A 34% BER decrease is reported compared with

conventional 1D-PDNP. [56] proposes to replace the APP detector with a DNN for a 3×3 multitrack detection. A window of 15 bits is cast on the equalized waveforms and outputs the probability of the current bit. The cross entropy loss is used for optimizing. Three neural networks are designed as follows. The first one is a fully-connected DNN. The network has five hidden fully-connected layers. The second neural network has five convolutional neural network layers. Each layer consists of a 2D convolutional layer, a batch normalization layer and a rectified linear unit (ReLU) activation function. The third network is a many-to-one recurrent neural network (RNN). At each time step, the input will go through seven stacks of bidirectional long short-term memory (LSTM) layer. Among all the three networks, CNN achieves the lowest BER and is chosen to plug in iterative (turbo) decoding. Compared with a standard BCJR-based 1D-PDNP, a 21.72% density gain is reported.

CHAPTER 2
MULTITRACK DETECTION WITH PATTERN DEPENDENT NOISE PREDICTION

2.1 Introduction

One goal of this chapter is to propose a multitrack detection framework that can detect multiple tracks with multiple readers. We need to redesign an equalizer and an associated target. We have shown the superior performance of 1D-PDNP in Section 1.3.2, but little work has been done to extend it to multitrack detection. Another object is to develop a multitrack detector with multitrack pattern-dependent noise prediction (2D-PDNP) to mitigate the 2D media noise. Besides, methods for training the parameters involved in the equalizer and the detector are also needed. We will test the structure and these algorithms on a set of quasi-micromagnetic simulated channel waveforms and compare the performance with a non-PDNP multitrack Viterbi detector.

This chapter is organized as follows. In Section 2.2, we propose the joint design of a multiple-input and multiple-output (MIMO) equalizer and a matrix-valued target for multitrack detection. In Section 2.3, we develop the multitrack pattern-dependent noise prediction Viterbi algorithm. In Section 2.4, we provide different methods to train the PDNP parameters for the multitrack detectors. In Section 2.6, we summarize this chapter.

2.2 Joint MMSE Optimization of Equalizer and Target

We consider the multitrack detection problem in which a set of M adjacent tracks are detected jointly, based on the observations from an array of N readers. To simplify notation we focus throughout this section on the case of detecting $M = 2$ tracks. Generalizing to $M > 2$ is straightforward. An example with two readers and two tracks is shown in Fig. 2.1.

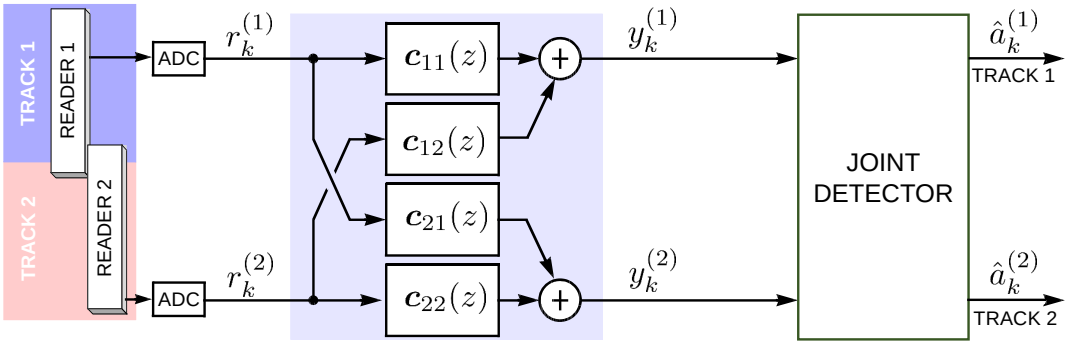


Figure 2.1: A multitrack detection example, where two readers ($N = 2$) are used to detect two tracks ($M = 2$).

Each reader produces a readback waveform with ISI, ITI, media noise, and electronic noise. Each readback waveform is sampled by an ADC, and then passed as an input to an N-input M-output equalizer. The equalized signal will be fed into a joint detector. We do not explicitly model any modulation or error-control coding. We assume that the tracks being detected are synchronous, both with each other and with the ADC sampling rate.

Here we summarize the MIMO equalizer and target that jointly minimize the equalizer output MSE, subject to a monic and minimum-phase constraint on the target [25].

The $M \times 1$ vector-valued equalizer output can be written compactly as $\mathbf{y}_k = \mathbf{C}^T \underline{\mathbf{r}}_k$, where $\underline{\mathbf{r}}_k = [\mathbf{r}_k^T, ..., \mathbf{r}_{k-N_c+1}^T]^T$ collects the relevant readback samples into a single vector, where $\mathbf{r}_k = [r_k^{(1)}, ..., r_k^{(N)}]^T$ is the k-th vector-valued input to the equalizer, and where $r_k{}^{(i)}$ is the k-th sample from the i-th reader, and where $\mathbf{C} = [\mathbf{C}_0^T, ..., \mathbf{C}_{N_c-1}^T]^T$ represents the MIMO equalizer, where each $\mathbf{C}_k$ is a $M \times N$ matrix-valued coefficient. The typical aim of such an equalizer is to transform the channel so that it matches a target, which in this case is a MIMO filter with transfer function $\mathbf{G}(D) = \sum_{k=0}^{\mu} \mathbf{G}_k D^k$, where μ is the target memory parameter.

To avoid a trivial solution, we impose the constraint that $\mathbf{G}_0$ is lower triangular with

ones on the diagonal [41], namely ($M = N = 2$)

$$\mathbf{G}_0 = \begin{bmatrix} 1 & 0 \\ G_{2,1}^{(0)} & 1 \end{bmatrix}. \tag{2.1}$$

This constraint can be viewed as the generalization of the monic constraint in the single-track case[57] to the MIMO case. We can view it as a set of four scalar-valued target filters instead of a single matrix-valued filter, where the i-th row and j-th column $G_{ij}(D)$ of $\mathbf{G}(D)$ represents the scalar target from the j-th track of interest to the i-th equalizer output ($i, j \in \{1, 2\}$). In terms of these scalar targets, the monic constraint of (2.1) implies that both $G_{11}(D)$ and $G_{22}(D)$ are monic in the scalar sense, and further that $G_{12}(D)$ is strictly causal. There are alternatives to this monic constraint that may also perform well. Other forms of a monic constraint for $\mathbf{G}_0$ can be found in [31, 32].

Filtering the vector sequence $\mathbf{a}_k = [a_k^{(1)}, a_k^{(2)}]^T \in \{\pm 1\}^2$ of information bits by the target results in the signal $\mathbf{s}_k = \mathbf{G}_0\mathbf{a}_{k-d} + \mathbf{B}^T\underline{\mathbf{a}}_k$, where d is the equalizer delay parameter, where $\underline{\mathbf{a}}_k = [\mathbf{a}_{k-d-1}^T, ... \ \mathbf{a}_{k-d-\mu}^T]^T$, and where we have introduced the target "tail" $\mathbf{B} = [\mathbf{G}_1, ..., \mathbf{G}_\mu]^T$. In these terms, the $M \times 1$ equalizer error vector at time k can be written as:

$$\mathbf{e}_k = \mathbf{C}^T\underline{\mathbf{r}}_k - \mathbf{G}_0\mathbf{a}_{k-d} - \mathbf{B}^T\underline{\mathbf{a}}_k.$$

The MMSE optimization problem is to jointly choose the equalizer $\mathbf{C}$ and target $(G_{2,1}^{(0)}, \mathbf{B})$ to minimize the mean-squared error $E(\|\mathbf{e}\|^2)$. This is the multitrack (MIMO) version of the GPR strategy for single-track detection that is reviewed in Section 1.3.1. The closed-form solution for the equalizer and target is:

$$\begin{aligned} [\mathbf{C}^T, -\mathbf{B}^T] &= \mathbf{M}^{-1}\mathbf{R}_{\mathbf{va}}^T\mathbf{R}_{\mathbf{vv}}^{-1} \\ \mathbf{G}_0 &= \mathbf{M}^{-1}, \end{aligned} \tag{2.2}$$

where $\mathbf{R_{vv}} = E(\mathbf{v}_k\mathbf{v}_k^T)$, $\mathbf{R_{va}} = E(\mathbf{v}_k\mathbf{a}_{k-d}^T)$, $\mathbf{v}_k = [\underline{\mathbf{r}}_k^T, \underline{\mathbf{a}}_k^T]^T$ and $\mathbf{M}$ is the monic factor in the Choleskey decomposition $\mathbf{I} - \mathbf{R}_{\mathbf{va}}^T\mathbf{R}_{\mathbf{vv}}^{-1}\mathbf{R}_{\mathbf{va}} = \mathbf{M}\mathbf{D}^2\mathbf{M}^T$.

2.3 Multitrack Detection with Multitrack PDNP

In this section, we propose the multitrack PDNP Viterbi algorithm. This algorithm can be viewed as an extension of the 1D-PDNP Viterbi algorithm [21] to the case of multitrack patterns and MIMO noise prediction (both downtrack and crosstrack).

2.3.1 A Multitrack Pattern-Dependent and Autoregressive Model for Noise

The equalizer derived in the previous section ensures that its output is as close a match (in a minimum-MSE sense) to a linear convolution of the bit sequences from the two tracks and the matrix-valued target. Nevertheless, media noise effects such as nonlinear transition shift diminish the accuracy of the linear convolution model; instead, we adopt a nonlinear model for the equalizer output at time k:

$$\mathbf{y}_k = \mathbf{s}(\mathbf{A}_k) + \mathbf{n}_k(\mathbf{A}_k), \tag{2.3}$$

where both the signal $\mathbf{s}$ and the zero-mean noise $\mathbf{n}$ depend on the 2D *pattern*:

$$\mathbf{A}_k = [\mathbf{a}_{k+I},\ ...,\ \mathbf{a}_k,\ ..., \mathbf{a}_{k-J}]. \tag{2.4}$$

The pattern $\mathbf{A}_k$ at time k is a matrix with two rows containing all of the bits from the two tracks that contribute to the equalizer output at time k. If the noise were independent and the linear convolution model were accurate, then the signal component would reduce to $\mathbf{s} = \sum_{i=0}^{\mu} \mathbf{G}_i\mathbf{a}_{k-i}$, so that the pattern would consist of only $\mathbf{a}_k$ through $\mathbf{a}_{k-\mu}$. Instead, because of nonlinear effects and pattern-dependent noise memory, the pattern includes I future bits and J past bits for each track.

A key property of media noise is that its characteristics depend on the pattern of bits that

were written. For example, a pattern with more transitions (both downtrack and crosstrack) will naturally have more media noise than a pattern with no transitions. To capture this effect, we model the vector-valued noise $\mathbf{n}_k$ in (2.3) as a vector-valued pattern-dependent autoregressive (AR) Gaussian process with memory N_p, so that:

$$\mathbf{n}_k = \sum_{i=0}^{N_p} \mathbf{P}_i(\mathbf{A}_k)\mathbf{n}_{k-i} + \mathbf{\Lambda}(\mathbf{A}_k)\mathbf{u}_k, \tag{2.5}$$

where $\mathbf{u}_k \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ is a spatially and temporally white sequence of Gaussian noise vectors, where

$$\mathbf{\Lambda}(\mathbf{A}_k) = \begin{bmatrix} \sigma_1(\mathbf{A}_k) & 0 \\ 0 & \sigma_2(\mathbf{A}_k) \end{bmatrix} \tag{2.6}$$

is a pattern-dependent standard deviation matrix, and where $\{\mathbf{P}_i(\mathbf{A}_k)\}$ are the matrix-valued AR filter coefficients, also dependent on the pattern $\mathbf{A}_k$. The zero-th coefficient $\mathbf{P}_0$ is included in the AR filter in order to account for spatial correlation in the noise; to be realizable we require $\mathbf{P}_0$ to be strictly lower triangle, with zeros on and above the diagonal. This multitrack noise model is an extension of the single-track pattern-dependent noise model [21], while the coefficient $\mathbf{P}_0$ is unique in the multitrack case. A block diagram showing the AR noise model for the special case of $N_p = 4$ is shown in Fig. 2.2.

2.3.2 The Joint ML Solution to the Multitrack Detection Problem

The Viterbi algorithm is a general solution to the problem of detecting the state sequence of a finite-state machine based on observations of its output in the face of independent noise [18]. The proposed model for the equalizer output fits this description precisely; as illustrated in Fig. 2.2, the state of the finite-state machine is

$$\boldsymbol{\theta}_k = [\mathbf{a}_{k+I-1}, \mathbf{a}_{k-1}, ..., \mathbf{a}_{k-N_p-J}]. \tag{2.7}$$

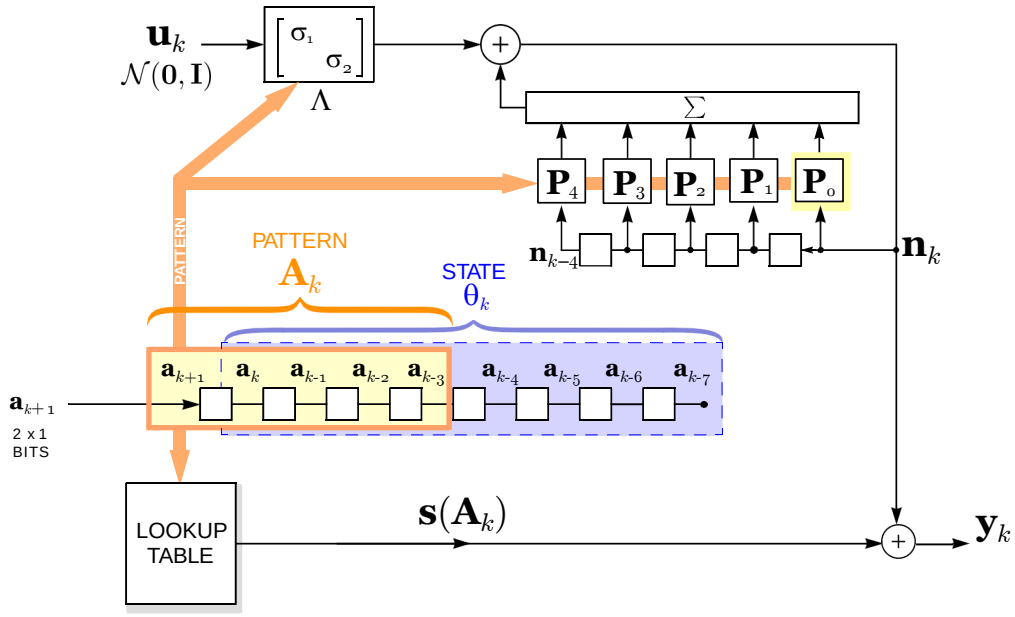


Figure 2.2: The signal and 2D media noise model for the special case of $N_p = 4$, $I = 1$, and $J = 3$.

The total number of states is $Q = 4^{N_p+I+J}$. Observe that the state $\boldsymbol{\theta}_k$ at time k uniquely determines the previous N_p patterns $\mathbf{A}_{k-1}$ through $\mathbf{A}_{k-N_p}$. Furthermore, the state $\boldsymbol{\theta}_k$ along with knowledge of the next pair of input bits $\mathbf{a}_{k+1}$ uniquely determines both the current pattern $\mathbf{A}_k$ and the next state $\boldsymbol{\theta}_{k+1}$.

We now derive the solution to the joint maximum-likelihood multitrack detector in the face of pattern-dependent AR Gaussian noise following the derivation for the 1D case [21]. Let L denote the length of the bit sequence on each track being detected, and let $L' = L + I + J + N_p$ denote the number of stages in the corresponding trellis diagram. Let $\underline{\mathbf{y}} = [\mathbf{y}_1, , ...\mathbf{y}_{L'}]$ be the observation sequence, and let $\underline{\boldsymbol{\theta}} = [\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_{L'+1}]$ be the state sequence to be estimated (which is equivalent to estimating the data written on the two tracks of interest). The ML solution chooses $\underline{\boldsymbol{\theta}}$ to maximize the likelihood function $f(\underline{\mathbf{y}} \mid \underline{\boldsymbol{\theta}})$, which

because of the noise model in Section 2.3.1 can be factored according to

$$\begin{aligned} f(\underline{\mathbf{y}} \mid \underline{\boldsymbol{\theta}}) &= f(\mathbf{y}_1 \mid \underline{\boldsymbol{\theta}}) \prod_{k=2}^{L'} f(\mathbf{y}_k \mid \mathbf{y}_1, ..., \mathbf{y}_{k-1}, \underline{\boldsymbol{\theta}}) \\ &= \prod_{k=1}^{L'} f(\mathbf{y}_k \mid \mathbf{Y}_k, \boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1}), \end{aligned} \tag{2.8}$$

where $\mathbf{Y}_k = [\mathbf{y}_{k-1}, ..., \mathbf{y}_{k-N_p}]$ is the set of N_p previous observation vectors. In terms of the trellis representing all possible trajectories of the state sequence $\underline{\boldsymbol{\theta}}$, consisting of Q states and spanning L' stages, we can interpret the objective function in (2.8) as a path metric formed by multiplying the branch metrics for each branch in the path. In particular, if we denote the edge (branch) in the trellis from state $\boldsymbol{\theta}_k$ to state $\boldsymbol{\theta}_{k+1}$ by $e = (\boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1})$. Associated with each edge e is a unique sequence of $N_p + 1$ patterns $\{\mathbf{A}_{k-N_p}, ..., \mathbf{A}_k\}$. The branch metric for an edge e is then the factor $f(\mathbf{y}_k \mid \mathbf{Y}_k, \boldsymbol{\theta}_k, \boldsymbol{\theta}_{k+1})$, which can be expressed as

$$f(\mathbf{y}_k \mid \mathbf{Y}_k, e) = \frac{1}{2\pi|\boldsymbol{\Lambda}(\mathbf{A}_k)|} \exp(-0.5 \parallel \boldsymbol{\Lambda}^{-1}(\mathbf{A}_k) \sum_{i=0}^{N_p} \mathbf{B}_i(\mathbf{A}_k)(\mathbf{y}_{k-i} - \mathbf{s}(\mathbf{A}_{k-i})) \parallel^2), \tag{2.9}$$

where $\mathbf{B}_i = \delta_i \mathbf{I} - \mathbf{P}_i$ can be interpreted as the coefficients of a MIMO prediction-error filter. Taking the negative logarithm of (2.9) and canceling constant terms that are common to all branches, we arrive at the additive branch metric

$$\gamma_k(e) = \log(\sigma_1^2(\mathbf{A}_k)\sigma_2^2(\mathbf{A}_k)) + \parallel \boldsymbol{\Lambda}^{-1}(\mathbf{A}_k) \sum_{i=0}^{N_p} \mathbf{B}_i(\mathbf{A}_k)(\mathbf{y}_{k-i} - \mathbf{s}(\mathbf{A}_{k-i})) \parallel^2, \tag{2.10}$$

where σ_1 and σ_2 are the diagonal components of $\boldsymbol{\Lambda}$.

The branch metric in (2.10) can be interpreted as implementing pattern-dependent noise prediction, but it is worth emphasizing that we did not set out with the goal of predicting noise in the first place; rather, it is just a byproduct of the ML solution in the face of our AR model for the noise. The noise prediction that occurs in (2.10) has two components: a

temporal component, where past noise samples are used to predict current noise samples, and a *spatial* component, where noise at one output of the equalizer is used to predict noise at another output.

The Viterbi algorithm can be used as an efficient solution to the problem of finding the path through the trellis with the smallest path metric. The pseudocode of the proposed multitrack PDNP Viterbi algorithm is shown in Algorithm 1, where $\Phi_k(q)$ is the partial path metric of state q at time k and $\pi_k(q)$ is the surviving state toward state q at time k. The trellis is set to terminate at state 0.

Algorithm 1 Multitrack Viterbi detection with 2D-PDNP

Input: Equalizer outputs $\{\mathbf{y}_k\},\ \mathbf{s}(\mathbf{A}), \mathbf{\Lambda}(\mathbf{A}), \mathbf{B}(\mathbf{A})\ \forall \mathbf{A}$
Output: $\hat{\mathbf{a}}_1, ...\hat{\mathbf{a}}_L$

$\Phi_1(0) = 0,\ \Phi_1(p) = \infty\ \forall p \neq 0$
for $k = 1$ to L' **do**
  for $q = 0$ to $Q - 1$ **do**
    for $p \in$ predecessors(q) **do**
      (p, q) determines $\{\mathbf{A}_{k-N_p}, ..., \mathbf{A}_k\}$
      $\gamma_k(p, q) = \log(\sigma_1^2(\mathbf{A}_k)\sigma_2^2(\mathbf{A}_k))$
      $+ \parallel \mathbf{\Lambda}^{-1}(\mathbf{A}_k) \sum_{i=0}^{N_p} \mathbf{B}_i(\mathbf{A}_k)(\mathbf{y}_{k-i} - \mathbf{s}(\mathbf{A}_{k-i})) \parallel^2$
      $p^* = \text{argmin}_p\{\Phi_k(p) + \gamma_k(p, q)\}$
      $\Phi_{k+1}(q) = \Phi_k(p^*) + \gamma_k(p^*, q)$
      $\pi_{k+1}(q) = p^*$
    end for
  end for
end for
Extract $\{\hat{\mathbf{a}}_1, ..., \hat{\mathbf{a}}_L\}$ from survivor minimizing $\Phi_{L'+1}(0)$

2.4 Estimation of PDNP Parameters

The proposed multitrack detector described in the previous section requires complete knowledge of the channel model; in particular, it requires knowledge of the pattern-dependent AR feedback filter coefficients $\{\mathbf{P}_i\}$, the pattern-dependent standard deviation matrices $\{\mathbf{\Lambda}\}$, and the pattern-dependent expected equalizer outputs $\{\mathbf{s}\}$. In practice these parameters would have to be estimated through a process we refer to as

training.

2.4.1 Non-Pattern-Dependent Multitrack Noise Prediction

With the equalizer fixed, we turn our attention to the remaining parameters $\mathbf{s}$, $\{\mathbf{P}_i\}$, and $\mathbf{\Lambda}$. In the special case when these parameters are time-invariant (independent of the pattern), the equalizer output reduces to a *stationary* vector-valued AR process with nonzero mean $\mathbf{s}$. We can rephrase the parameter estimation problem as a linear prediction problem, because the prediction parameters $\{\hat{\mathbf{P}}_i\}$ and $\hat{\mathbf{s}}$ that minimize the MSE

$$\text{MSE} = E(\parallel \mathbf{y}_k - \hat{\mathbf{s}} - \sum_{i=0}^{N_p} \hat{\mathbf{P}}_i(\mathbf{y}_{k-i} - \hat{\mathbf{s}}) \parallel^2), \tag{2.11}$$

subject to the causal constraint ($\mathbf{P}_0$ strictly lower triangular), are precisely the AR parameters $\mathbf{s}$ and $\{\mathbf{P}_i\}$, respectively [58].

The MMSE estimate for $\mathbf{s}$ is thus $\hat{\mathbf{s}} = E(\mathbf{y}_k)$, and the MMSE estimate for $\{\mathbf{P}_i\}$ is the set of prediction coefficients that minimize $E(\parallel \mathbf{n}_k - \hat{\mathbf{n}}_k \parallel^2)$, where $\hat{\mathbf{n}}_k = \sum_{i=0}^{N_p} \mathbf{P}_i \mathbf{n}_{k-i}$. In the case of scalar linear prediction, there would be no $\mathbf{P}_0$ term, but in the vector case it can exist as long as the overall linear prediction filter is strictly causal in the space-time sense [41]: here, this means that $\mathbf{P}_0$ is a strictly lower triangular matrix, with zeros on and above the diagonal.

In terms of the zero-th prediction-error filter coefficient $\mathbf{B}_0 = \mathbf{I} - \mathbf{P}_0$, the predictor "tail" $\tilde{\mathbf{P}} = \left[\mathbf{P}_1, \ ..., \ \mathbf{P}_{N_p}\right]$, and $\mathbf{N}_k = [\mathbf{n}_{k-1}^T, \ ..., \ \mathbf{n}_{k-N_p}^T]^T$, the autocorrelation matrix of the prediction error $\mathbf{e}_k = \mathbf{n}_k - \sum_{i=0}^{N_p} \mathbf{P}_i \mathbf{n}_{k-i}$ can be written as

$$\begin{aligned}
\mathbf{R_{ee}} &= E((\mathbf{B}_0\mathbf{n}_k - \tilde{\mathbf{P}}\mathbf{N}_k)(\mathbf{B}_0\mathbf{n}_k - \tilde{\mathbf{P}}\mathbf{N}_k)^T) \\
&= \mathbf{B}_0\mathbf{R_{nn}}\mathbf{B}_0^T + \tilde{\mathbf{P}}\mathbf{R_{NN}}\tilde{\mathbf{P}}^T - \mathbf{B}_0\mathbf{R_{Nn}}^T\tilde{\mathbf{P}}^T - \tilde{\mathbf{P}}\mathbf{R_{Nn}}\mathbf{B}_0^T \\
&= (\tilde{\mathbf{P}} - \mathbf{B}_0\mathbf{R_{Nn}}^T\mathbf{R_{NN}}^{-1})\mathbf{R_{NN}}(\tilde{\mathbf{P}} - \mathbf{B}_0\mathbf{R_{Nn}}^T\mathbf{R_{NN}}^{-1})^T \\
&+ \mathbf{B}_0(\mathbf{R_{nn}} - \mathbf{R_{Nn}}^T\mathbf{R_{NN}}^{-1}\mathbf{R_{Nn}})\mathbf{B}_0^T,
\end{aligned} \tag{2.12}$$

where $\mathbf{R_{nn}} = E(\mathbf{n}_k\mathbf{n}_k^T)$ is the autocorrelation matrix of the current noise vector, $\mathbf{R_{NN}} = E(\mathbf{N}_k\mathbf{N}_k^T)$ is the autocorrelation of the previous noise vectors, and $\mathbf{R_{Nn}} = E(\mathbf{N}_k\mathbf{n}_k^T)$ is the cross-correlation matrix between the current and previous noise vectors. To minimize $\mathrm{MSE} = \mathrm{tr}(\mathbf{R_{ee}})$, we can do no better than zeroing the first term in (2.12) by choosing $\tilde{\mathbf{P}} = \mathbf{B}_0\mathbf{R_{Nn}}^T\mathbf{R_{NN}}^{-1}$. In terms of the monic Cholesky decomposition:

$$\mathbf{R_{nn}} - \mathbf{R_{Nn}}^T\mathbf{R_{NN}}^{-1}\mathbf{R_{Nn}} = \mathbf{M}\mathbf{\Lambda}^2\mathbf{M}^T, \tag{2.13}$$

where $\mathbf{M}$ is monic (ones on the diagonal) and lower triangular, and $\mathbf{\Lambda}$ is nonnegative and diagonal, the trace of the remaining term in (2.12) is minimized when $\mathbf{B}_0\mathbf{M} = \mathbf{I}$ (a simple consequence of the fact that the product of two monic and lower triangular matrices is also monic and lower triangular), so that the optimal zero-th coefficient is $\mathbf{P}_0 = \mathbf{I} - \mathbf{M}^{-1}$. With this choice, the optimized autocorrelation matrix reduces to $\mathbf{R_{ee}} = \mathbf{\Lambda}^2$. To summarize, for the special case when the parameters are independent of the pattern, the MMSE estimates of the PDNP parameters can be written as

$$\begin{aligned}
&\mathbf{s} = E(\mathbf{y}_k), \\
&\mathbf{P}_0 = \mathbf{I} - \mathbf{M}^{-1}, \\
&[\mathbf{P}_1,\ ...,\ \mathbf{P}_{N_p}] = \mathbf{M}^{-1}\mathbf{R_{Nn}}^T\mathbf{R_{NN}}^{-1},
\end{aligned} \tag{2.14}$$

where $\mathbf{M}$ and $\mathbf{\Lambda}$ are the factors specified by the monic Cholesky decomposition of (2.13).

2.4.2 Estimating AR Channel Parameters via Training

In this subsection, we drop the assumption that the parameters are independent of the pattern, and describe a training process for estimating the parameters of the pattern-dependent AR feedback filter coefficients $\{\mathbf{P}_i(\mathbf{A})\}$, the pattern-dependent standard deviation matrices $\{\mathbf{\Lambda}(\mathbf{A})\}$, and the pattern-dependent expected equalizer outputs $\{\mathbf{s}(\mathbf{A})\}$ (The equalizer remains independent of the pattern.) The procedure is

straightforward, and requires that a training sequence of L_t known bits be written on each track, separately from the user data. At time k in the training process, when reading back the training sequence, the pattern $\mathbf{A}_k$ will be known. We can thus organize the readback data according to the known pattern, and separately implement the linear predictors from the previous section for each pattern. In this way, each pattern will have its own expected signals, predictors, and standard deviation matrix. According to the definition in (2.4), the total number of patterns is 4^{I+J+1}. Suppose we have a basket for each such pattern. The training steps are as follows.

Step 1 *First pass*: For each $k \in \{1, ...L_t\}$, place $\mathbf{y}_k$ into basket $\mathbf{A}_k$;

Step 2 For each basket $\mathbf{A}$, average its $\mathbf{y}_k$ vectors to estimate $\mathbf{s}(\mathbf{A})$.

Step 3 Empty all baskets.

Step 4 *Second pass*: For each $k \in \{1, ...L_t\}$, place $\mathbf{n}_k = \mathbf{y}_k - \mathbf{s}(\mathbf{A}_k)$ and

$\mathbf{N}_k = [\mathbf{n}_{k-1}^T,\ ...,\ \mathbf{n}_{k-N_p}^T]^T$ in the basket $\mathbf{A}_k$;

Step 5 For each basket $\mathbf{A}$, estimate

$$\mathbf{R_{nn}} = E(\mathbf{n}_k \mathbf{n}_k^T)$$

$$\mathbf{R_{NN}} = E(\mathbf{N}_k \mathbf{N}_k^T)$$

$$\mathbf{R_{Nn}} = E(\mathbf{N}_k \mathbf{n}_k^T)$$

Step 6 For each basket, perform the Cholesky decomposition

$$\mathbf{R_{nn}} - \mathbf{R_{Nn}}^T \mathbf{R_{NN}}^{-1} \mathbf{R_{Nn}} = \mathbf{M} \mathbf{\Lambda}^2 \mathbf{M}^T$$

Step 7 For each basket, the predictor coefficient matrices are obtained by $\mathbf{P}_0 = \mathbf{I} - \mathbf{M}^{-1}$

$$[\mathbf{P}_1,\ ...,\ \mathbf{P}_{N_p}] = \mathbf{M}^{-1} \mathbf{R_{Nn}}^T \mathbf{R_{NN}}^{-1}$$

The standard deviation matrix of the prediction error is $\mathbf{\Lambda}$.

2.4.3 Self Training Scheme for PDNP

Both 1D and 2D PDNP Viterbi algorithms require accurate estimates of the parameters (including mean offsets, predictor coefficients or matrices, and residual variances or covariance matrices) of the noise model. These parameters can be estimated through a training process as described in the previous section, based on the readback waveforms from a known set of training bits, with the expectation that the noise behavior learned during training will be applicable when later reading back a sector of unknown bits. In this subsection, we propose a new "training" strategy called *self training* that eliminates the need for any a priori training. Instead, all parameters can be estimated from scratch, independently from one sector to the next, through an iterative process that iterates between a soft-output channel detector and a channel model estimator. No knowledge of any training bits is required. Importantly, unlike the traditionally trained case (in which a different set of bits on a different part of the disk are used to estimate the parameters), the parameters estimated via self training are based on the same bits, written on the same part of the disk, as those being detected.

The self training scheme is shown in Fig. 2.3. Similar to the expectation maximization (EM) algorithm, the rough idea is to iterate between a PDNP detector (that assumes that the model it receives is accurate) and a model estimator (that assumes the LLR signs are reliable). The hope is that, as iterations progress, the LLR's will grow more reliable, which will in turn result in a better estimate of the model, which will in turn lead to even more reliable LLR's, and so on. As shown in the figure, the readback waveform is first equalized and then fed to a BCJR non-PDNP detector, which produces LLR's for the bits. The training proceeds in the traditional way, with two exceptions: the signs of the LLR's are used in place of training bits, and only when all of the LLR's in a block exceed a threshold in magnitude is the data used at all, for in this case the bit decisions are expected to be reliable. The estimated model parameters are then fed back to the detector for the next iteration of the PDNP detector. This process can be iterated several times if

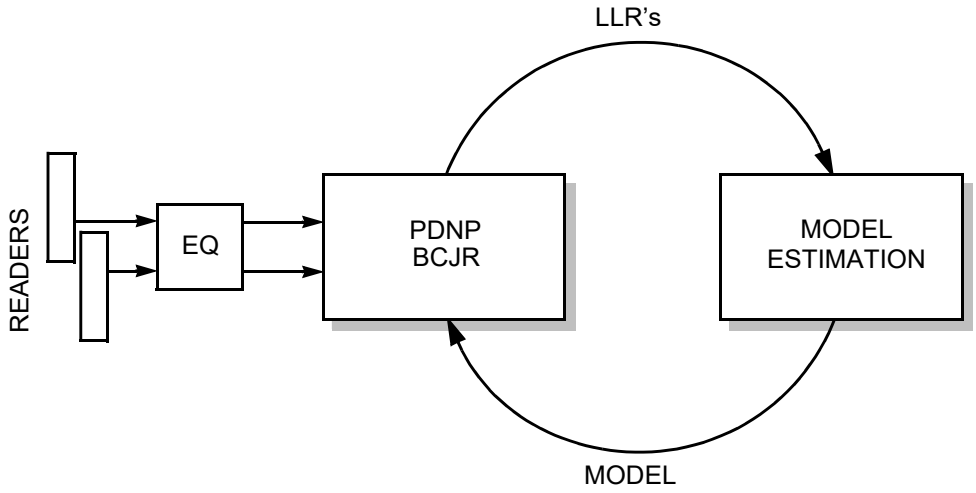


Figure 2.3: The self training scheme.

necessary.

2.5 Quantitative Results

2.5.1 Ehime Waveforms

We test our algorithm on simulated waveforms provided by Ehime University, which were produced by realistic head fields and a Voronoi medium with Stoner-Wohlfarth switching [30]. Five consecutive tracks are written in a shingled fashion, as shown in Fig. 2.4. In each track, there are 40950 independent pseudorandom bits sequence and 128 bits preamble and postamble, respectively. A total of 900 readback waveforms are produced with a fixed bit length of 7.3 nm, track pitches from 16.1 nm to 26.1 nm (2 nm increment), reader widths from 70% to 145% (15% increment) of a nominal reader width and 25 reader positions (spanning from the second to fourth tracks at one-eighth of a track increment). Readback waveforms from different tracks are perfectly synchronized and no modulation coding or error-control coding is applied.

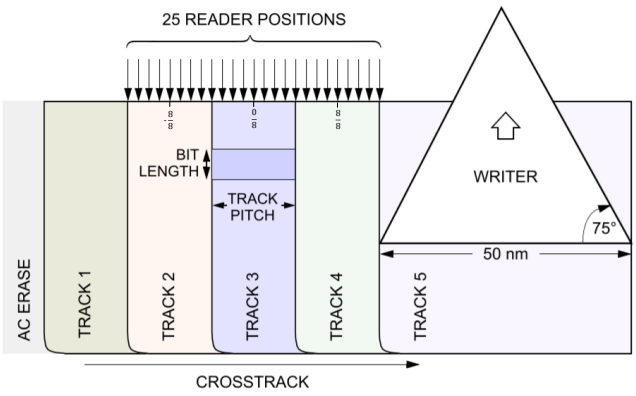


Figure 2.4: Geometry of the Ehime waveform model, showing the five-shingled tracks and 25 possible reader locations [30].

2.5.2 PDNP Parameters via Training

In this simulation, no electronic noise is added. We set the number of equalizer coefficients $N_c = 11$ and the monic target memory $\mu = 1$ for the following simulations. We train parameters with samples from track two and three and then test 2D-PDNP on track three and four.

We first compare the performance of multitrack Viterbi detector (without any noise prediction) to the proposed multitrack PDNP Viterbi detector. Without noise prediction, the trellis has only four states and its branch metric is the squared Euclidean distance between the equalizer output and the target output (when fed by the bits that correspond to the branch). We test four-bit PDNP ($I = 1, J = 0$) and six-bit PDNP ($I = J = 1$), and there are only two prediction matrices $\mathbf{P}_0$ and $\mathbf{P}_1$ ($N_p = 1$), so that the number of states with multitrack PDNP is 16 and 64, respectively. We train the equalizer and PDNP parameters on track two and three, and test on track three and four.

The results are shown in Fig. 2.5, where we plot BER (averaged over the two tracks) versus track pitch. Every parameter (reader width, reader location, equalizer, and target) is separately optimized for each point in the curve so as to minimize the resulting BER. The upper blue curve shows the performance of multitrack Viterbi without noise prediction,

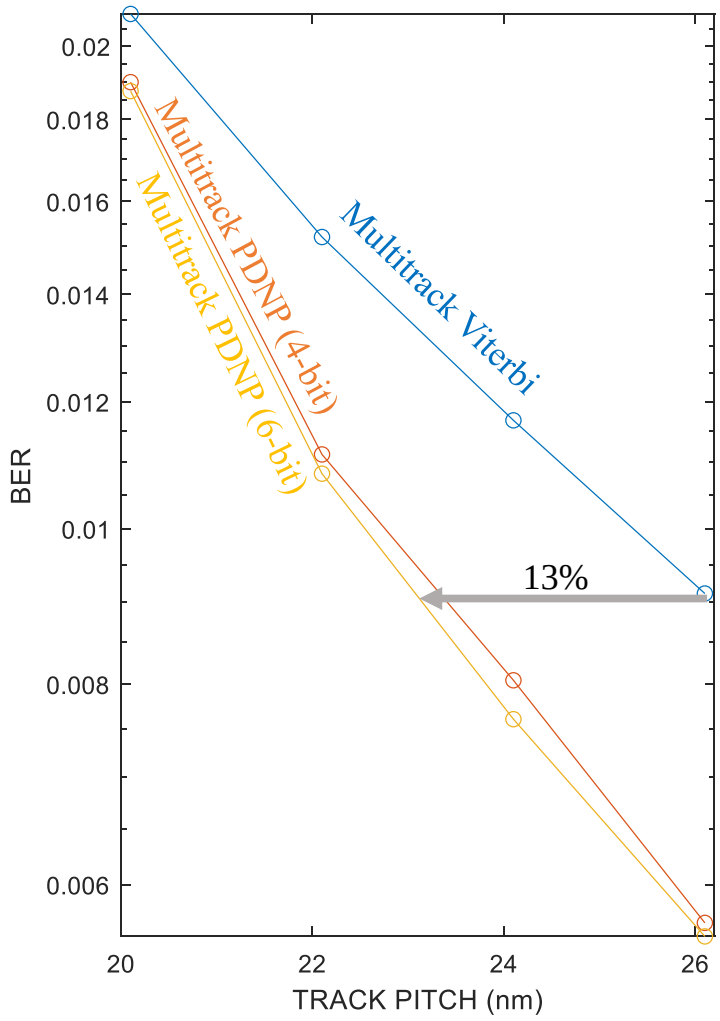


Figure 2.5: BER performance of multitrack Viterbi, multitrack PDNP (4-bit) and multitrack PDNP (6-bit).

while the lower two curves show performance with four-bit and six-bit PDNP.

The optimal reader width with multitrack PDNP is 85% for each reader at track pitch 26.1nm, 70% at track pitch 20.1, 22.1 nm and 24.1 nm. The optimal reader width without multitrack PDNP is 85% for reader one and 100% for reader two at track pitch 26.1 nm, 100% for reader one and 85% for reader two at track pitch 24.1 nm, 85% for reader one and 70% for reader two at track pitch 22.1 nm, and 70% for each reader at the track pitch 20.1 nm. The optimal reader positions are centered over the tracks of interest for all curves.

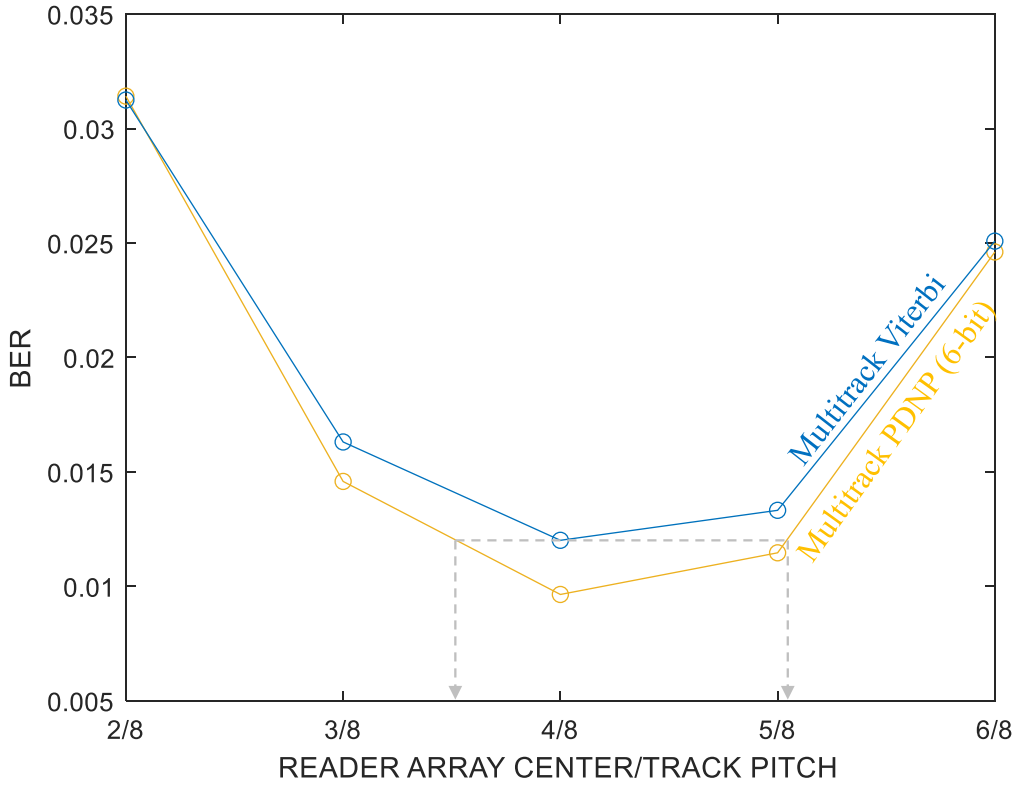


Figure 2.6: The bathtub curves with multitrack Viterbi and multitrack PDNP (6-bit).

At the BER of 0.9%, the benefit of multitrack PDNP is seen to lead to a roughly 13% gain in areal density. We did not consider pattern lengths longer than six bits because that was not sufficient training data for each pattern. Generally speaking, multitrack PDNP requires narrower read width and centered position to reach its optimal BER.

However, readers with narrower width are more challenging to manufacture and the actuator arm cannot guarantee the reader is always centered on a track. We next investigate the influence of reader positions on multitrack Viterbi detection and the proposed multitrack PDNP Viterbi detector. The waveforms are from the dataset with track pitch of 26.1 nm, 130% reader width and normalized reader positions from -1 to 0 in steps of one-eighth of a track pitch. The distance between the two readers is fixed to the half of the track pitch. The results are shown in Fig. 2.6, where we plot the BER versus the reader array center. The upper blue "bathtub" curve shows the BER of the multitrack Viterbi detection, while the

yellow bottom curve shows the BER of the proposed multitrack PDNP Viterbi algorithm with six-bit patterns ($I = J = 1$). The plot clearly shows the proposed PDNP Viterbi algorithm has a wider bathtub bottom, which means it is less sensitive to reader positions than multitrack Viterbi detection.

2.5.3 PDNP Parameters via Self Training

We test the self training algorithm using the same database of quasi-micromagnetic simulated waveforms from Ehime University. Independent white Gaussian electronic noise with zero mean and standard deviation $\sigma_e = 0.04$ was added to each twice oversampled readback sample. The architecture in Fig. 2.3 is tested with an MMSE equalizer with 22 fractionally spaced coefficients and the central track with track pitch 24.1 nm is detected. For the first pass, the PDNP parameters are initialized to "zero" so that straight BCJR is executed. We present results for 1D-PDNP with a pattern length of three bits and two predictor coefficients. Fig. 2.7 shows the BER after the second iteration of the PDNP BCJR versus threshold used in the second model estimation. The optimal threshold is a trade-off between the reliable decisions enabled by a large threshold and the quantity of data enabled by a small threshold.

A plot of BER versus iteration number is shown in Fig. 2.8 with the track pitch of 26.1 nm and the threshold of six. Interestingly, here we see that the self training algorithm outperforms a genie-aided training (which somehow has perfect knowledge of the bits for model estimation purposes only) at the fourth iteration. These results indicate that the genie-aided training that minimizes the MMSE does not necessarily minimize the BER. The fact that MMSE parameters are not optimal with respect to minimizing BER inspires us to seek better parameters that we will explore in the next chapter. We also test the BER versus track pitch after iterations and optimal reader width and position as well as threshold. Further experiment results show that self training algorithm gives the same performance as genie-aided training, which is a 4% increase in areal density over a non-PDNP Viterbi

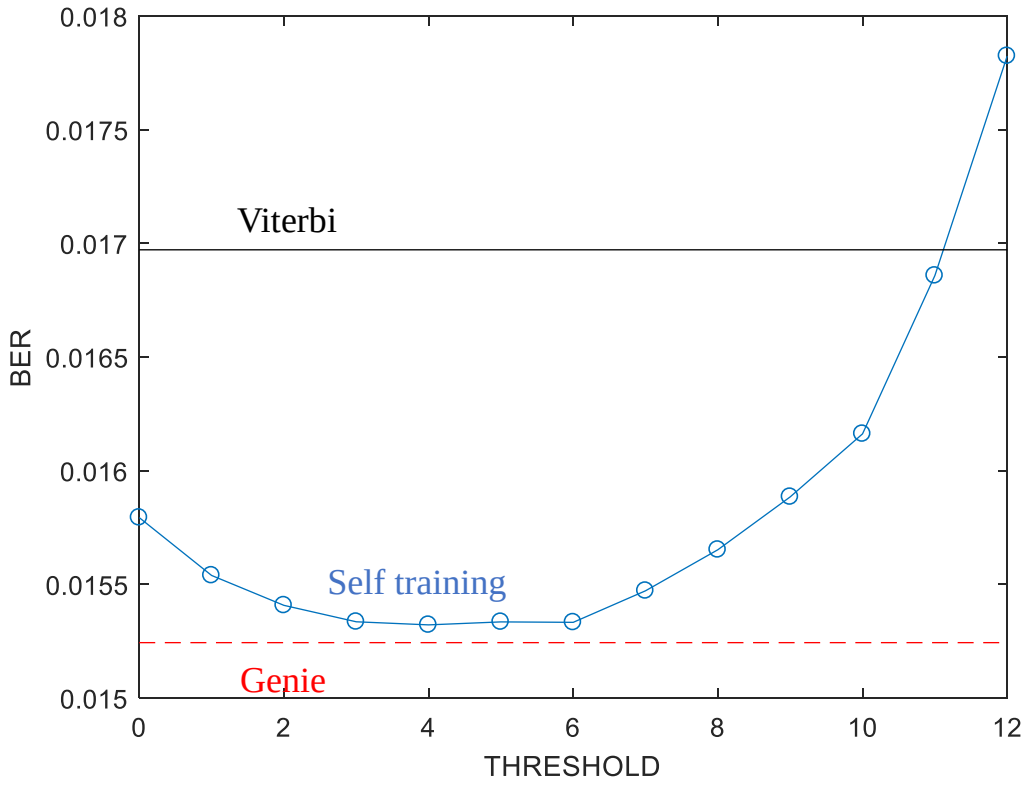


Figure 2.7: BER versus threshold at track pitch 24.1 nm in the second model estimation.

detector.

This experiment can also be extended and applied to multitrack detection. The equalizer is replaced with a MIMO equalizer and the 1D-PDNP BCJR detector is extended to the multitrack BCJR detector with 2D-PDNP presented in Section. 2.3. The model estimation follows the training process depicted in Section. 2.4, but the training bits are replaced with the sign of LLR's above a threshold.

2.6 Summary

In this chapter, we propose a multitrack detection framework that can detect multiple tracks with multiple readers. Under this framework, we propose the multitrack detector with 2D-PDNP, which solves the joint maximum-likelihood sequence detection problem

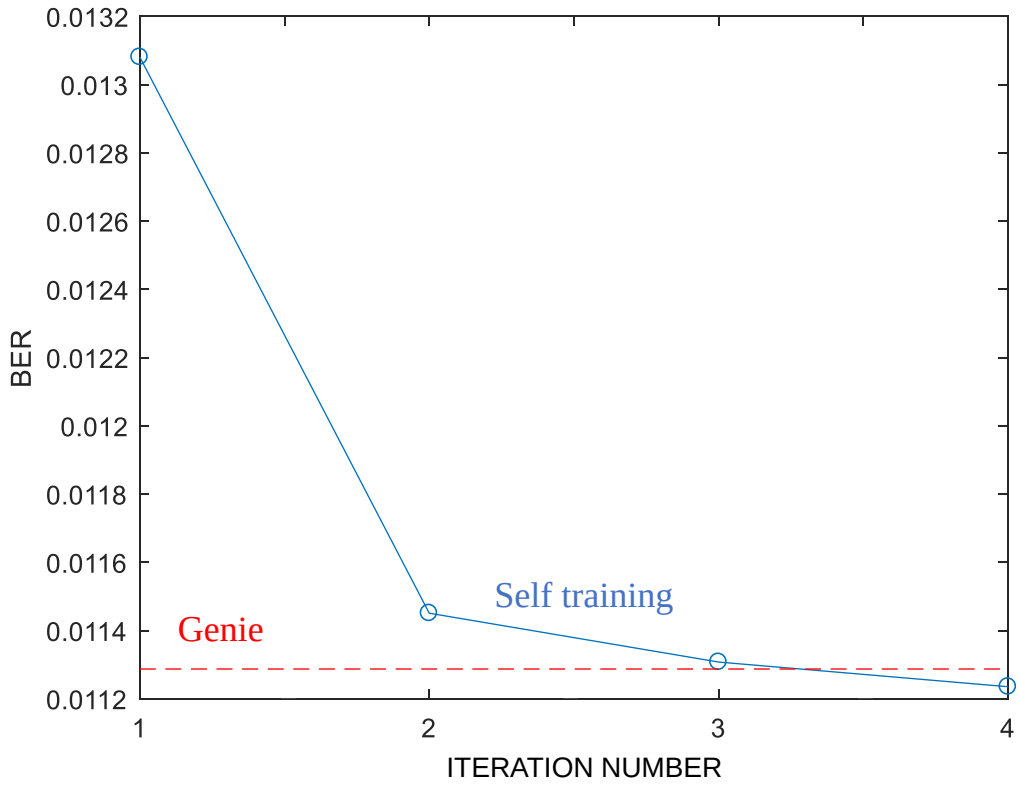


Figure 2.8: BER evolution of self training algorithm at track pitch 26.1 nm and threshold 6.

for multitrack detection when the noise is pattern-dependent AR and Gaussian. In contrast to prior work, our pattern-dependent noise predictor is matrix-valued, so that it implements both crosstrack and downtrack noise prediction, taking into account transitions occurring in both downtrack and crosstrack directions. We propose two methods for training the parameters of PDNP, one based on a minimum MSE criterion and the other based on a self-training scheme. We also apply these methods to a set of quasi-micromagnetic simulated channel waveforms. Numerical results show 13% areal density gain is achieved by multitrack PDNP over multitrack Viterbi.

From the next chapter forward, we will keep using the multitrack detection structure proposed in this chapter. In Chapter 3, we optimize the PDNP parameters aiming at minimizing the BER after detection. Later in Chapter 4, we bring decoding into the

picture and optimize the PDNP parameters to minimize the frame-error rate.

CHAPTER 3
MINIMUM-BIT-ERROR RATE TUNING

3.1 Introduction

Traditionally, the PDNP parameters are chosen to minimize some form of MMSE criterion like what we did in Chapter 2, due to its quadratic form which ensures a closed-form solution and a single local minimum. However, the MMSE parameters do not necessarily minimize the BER as we demonstrated in Section 2.4.3, which is the more relevant performance metric from the perspective of the end user interested in maximizing areal density.

Furthermore, moving from 1D to 2D PDNP detection results in an explosion in the number of detector parameters, primarily because the number of patterns grows exponentially in the number of tracks being detected, and further because some of the parameters (like equalizer coefficients and predictor coefficients) become matrix-valued instead of scalars. A closed-form solution for the MMSE parameters requires full knowledge of the noise second-order statistics for each pattern, which can be impractical for a multitrack detector, where the number of patterns is large and may exceed the amount of training data available. The job of this chapter is to examine the question of how to choose the parameters of a PDNP detector and an equalizer in order to minimize the BER. The key step is to find a reasonable cost function since the BER of a sequence detector cannot be minimized directly. The solution also needs to fit in the proposed detection structure.

In this chapter, we develop the adaptive minimum-frame-error rate algorithm in Section 3.2, in the context of single-track detection, extend it to multitrack detection, and explore its properties in Section 3.3. In Section 3.4, we evaluate the performance of the

proposed algorithm using simulated waveforms and explore the optimization for parameter space. In Section 3.5, we summarize this chapter.

3.2 The Cost Function and AMBER Algorithm

We consider the problem of adapting the parameters of a typical magnetic recording read channel, such as those illustrated in Fig. 3.1. For example, consider the 1D scenario depicted in Fig. 3.1(a), where the readback waveform from a single reader is sampled, equalized, and fed to a 1D trellis-based detector, which ultimately results in a sequence of decisions $\hat{a}_k$ about the written bits a_k. In this case, the parameters to be adapted are the coefficients of the equalizer along with any parameters within the detector. As another example, we consider the multitrack scenario depicted in Fig. 3.1(b), where multiple readback waveforms are sampled and equalized by a MIMO equalizer before being fed to a trellis-based multitrack detector, which produces decisions $\hat{\mathbf{a}}_k$ about the bits written on the multiple tracks. Throughout this chapter we will use $\mathbf{\Theta}$ to denote the set of read channel parameters that are to be optimized. In the case of a PDNP detector, $\mathbf{\Theta}$ includes the equalizer coefficients, the pattern-dependent signal levels, the pattern-dependent predictor coefficients, and the pattern-dependent residual variances.

Before we propose the algorithm, we will first introduce a closely related performance metric to BER, known as the *path metric margin*. Roughly speaking, the path metric margin at time k measures the gap between the metric of a competing path and that of the correct path. More precisely, assume that the written bits a_0 through a_k are known, as would arise during a training phase. In this case, if θ_k denotes the state at time k of a Viterbi detector, this implies that the correct path with state sequence $\{\theta_0, \ ... \ \theta_k\}$ leading to the correct state θ_k is known. We define the *competing* path at time k as the "best of the rest" of the paths that lead to state θ_k at time k; in particular, when the Viterbi algorithm aims to find the path with minimum metric, the competing path at time k is the partial path that leads to to the correct state at time k, excluding the correct path, with minimal metric. The competing path can

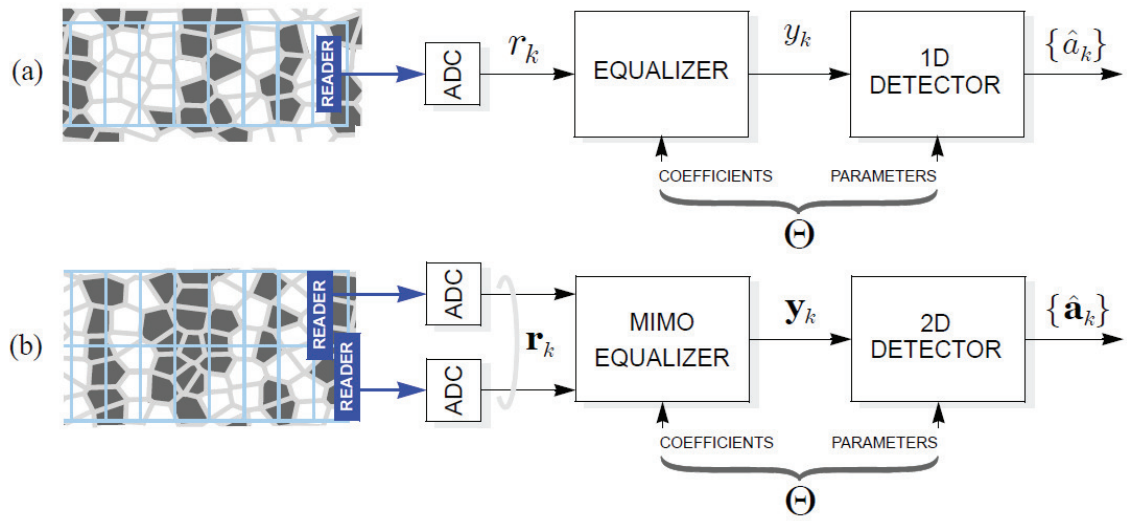


Figure 3.1: Illustration of the parameters to be optimized, for two scenarios: (a) for single-track detection using a single reader; (b) for multitrack detection using multiple readers.

be traced back from θ_k until it merges with the correct path, defining the competing path sequence $\{\hat{\theta}_{k-\ell_k}, \ldots \hat{\theta}_k\}$, where ℓ_k denotes the length of the separation between the correct and competing paths. Because the competing path starts and ends on the correct path, we have $\hat{\theta}_{k-\ell_k} = \theta_{k-\ell_k}$ and $\hat{\theta}_k = \theta_k$. The path metric margin is then simply the difference between the competing and correct path metrics:

$$M_k = \sum_{i=0}^{\ell_k-1} \gamma(\hat{\theta}_{k-i}, \hat{\theta}_{k-i-1}; \mathbf{\Theta}) - \sum_{i=0}^{\ell_k-1} \gamma(\theta_{k-i}, \theta_{k-i-1}; \mathbf{\Theta}), \tag{3.1}$$

where $\gamma(\theta_k, \theta_{k-1})$ is the branch metric from state θ_{k-1} to θ_k.

A large margin implies that the correct path is easily distinguishable from the incorrect path, while a small positive margin implies that the correct path is barely preferred over the incorrect path. A negative margin (when $M_k < 0$) implies that a Viterbi detector that ignores the training information would make bit errors, either at or near time k. Note that one-bit error can result in multiple negative margins. Based on this observation one might be tempted to choose the parameters $\mathbf{\Theta}$ to minimize the probability that M_k is negative, or

(in terms of the unit-step function) so as to minimize the following cost function:

$$J(\mathbf{\Theta}) = E(u(-M_k)), \tag{3.2}$$

where $E(\ \cdot\ )$ is the expectation and $u(\ \cdot\ )$ is the unit step function. However, this function is not differentiable and is difficult to minimize directly. Therefore, we instead propose to choose $\mathbf{\Theta}$ so as to minimize the following cost function:

$$J_\tau(\mathbf{\Theta}) = E((\tau - M_k)u(\tau - M_k)), \tag{3.3}$$

where τ is a small positive threshold. Not only is $J_\tau(\mathbf{\Theta})$ differentiable, but under certain circumstances (explained in the next section) minimizing $J_\tau(\mathbf{\Theta})$ is equivalent to minimizing $J(\mathbf{\Theta})$.

Applying the stochastic gradient algorithm to $J_\tau(\mathbf{\Theta})$ leads to the AMBER algorithm for adapting the parameters $\mathbf{\Theta}$:

$$\mathbf{\Theta}_{k+1} = \mathbf{\Theta}_k + \lambda u(\tau - M_k)\nabla_{\mathbf{\Theta}} M_k, \tag{3.4}$$

where λ is the step size. The unit step factor in (3.4) acts as an indicator function: it ensures that the parameters only update when $M_k < \tau$. In other words, the parameters update only when the margin is dangerously small; otherwise, the parameters do not change. This feature is in stark contrast to MMSE algorithms like LMS and recursive least squares (RLS), which would continually update the parameters.

The AMBER algorithm of (3.4) is general and can be applied to a wide range of detector architectures, in any specific application one must first find an expression for the path metric margin in terms of the detector parameters, so that the gradient in (3.4) can be computed explicitly. We close this section with three specific examples.

3.2.1 AMBER for Viterbi without PDNP

We first consider a 2^{μ}-state Viterbi detector without PDNP, where μ is the channel memory. In this case the only parameters to optimize are the equalizer coefficients and the *signal levels* associated with each state transition. The signal levels can be viewed as the entries of a look-up table, one signal level for each state transition. Each state transition can be represented by a vector or *pattern* $\mathbf{a}$ of $\mu + 1$ bits (containing the current input bit as well as the μ previous input bits). We use the notation $s(\mathbf{a})$ to denote the signal level associated with the bit pattern $\mathbf{a}$. In this case the path metric margin of (3.1) reduces to:

$$\begin{aligned} M_k &= \sum_{i=0}^{\ell_k - 1} (y_{k-i} - s(\hat{\mathbf{a}}_{k-i}))^2 - (y_{k-i} - s(\mathbf{a}_{k-i}))^2 \\ &= \sum_{i=0}^{\ell_k - 1} 2\mathbf{c}^T \mathbf{r}_{k-i}(s(\mathbf{a}_{k-i}) - s(\hat{\mathbf{a}}_{k-i})) + s^2(\hat{\mathbf{a}}_{k-i}) - s^2(\mathbf{a}_{k-i}), \end{aligned} \tag{3.5}$$

where $\mathbf{a}_k = [a_k, ..., a_{k-\mu}]^T$ is a vector of bits for the correct path, where $\hat{\mathbf{a}}_k$ are the corresponding bits for the competing path, and where $\mathbf{c}$ is the vector of equalizer coefficients and $\mathbf{r}_k$ is the vector of relevant waveform samples at time k. Differentiating M_k with respect to $\mathbf{c}$ and $\mathbf{s}$ and substituting into (3.4) leads to explicit AMBER update equations for $\mathbf{c}$ and $s(\,\cdot\,)$:

$$\begin{aligned} &\mathbf{c}_{k+1} = \mathbf{c}_k + \lambda u(\tau - M_k) \sum_{i=0}^{l-1} (s(\mathbf{a}_{k-i}) - s(\hat{\mathbf{a}}_{k-i}))\mathbf{r}_{k-i}, \\ &\text{for each } i \in \{0, 1, ..., l-1\} \\ &\qquad s(\mathbf{a}_{k-i})_{k+1} = s(\mathbf{a}_{k-i})_k + \lambda u(\tau - M_k)(\mathbf{c}^T \mathbf{r}_{k-i} - s(\mathbf{a}_{k-i})) \\ &\qquad s(\hat{\mathbf{a}}_{k-i})_{k+1} = s(\hat{\mathbf{a}}_{k-i})_k + \lambda u(\tau - M_k)(-\mathbf{c}^T \mathbf{r}_{k-i} + s(\hat{\mathbf{a}}_{k-i})). \end{aligned} \tag{3.6}$$

The update equation derived in this way for the equalizer is similar to the NMBER equalizer that was proposed in [42] with different choices of the step size and threshold.

Since the sign of M_k is invariant to scaling, there are infinitely many combinations of $\mathbf{c}$ and $\mathbf{s}$ that achieve the same BER. One way to ensure convergence and avoid numerical

finite-precision effects is to normalize the equalizer vector after each update so that it has the same norm.

3.2.2 AMBER for 1D-PDNP

As a second example, consider the AMBER algorithm applied to a single-track PDNP detector, so that the parameters $\mathbf{\Theta}$ to be optimized are the equalizer $\mathbf{c}$, the signal levels $s(\mathbf{a})$, the residual prediction-error variances $v(\mathbf{a})$, and the $N_p \times 1$ predictor coefficient vectors $\mathbf{p}(\mathbf{a})$ associated with each bit pattern $\mathbf{a}$. The functional notation for these parameters will simplify our presentation, but in practice these parameters can be implemented using one large lookup table, where each row corresponds to a different bit pattern $\mathbf{a}$, and the first column represents the corresponding signal level, the second column represents the corresponding noise variance, and the remaining columns represent the prediction coefficients. The branch metric for an edge (branch) $e_k = (\theta_{k-1}, \theta_k)$ connecting state θ_{k-1} at time $k-1$ to state θ_k at time k for the PDNP detector, given the equalizer output vector $\mathbf{y}$, can be expressed as [21]

$$\gamma_k(e_k) = \ln v(e_k) + \frac{([1,\ -\mathbf{p}^T(e_k)](\mathbf{y}_k^{k-N_p} - \mathbf{s}(e_k)))^2}{v(e_k)}, \tag{3.7}$$

where $\mathbf{y}_k^{k-N_p} = [y_k, y_{k-1}, ..., y_{k-N_p}]^T$ and $y_k = \mathbf{c}^T[r_k, r_{k-1}, ..., r_{k-N_c}]^T$, where $\mathbf{s}(e_k) = [s(a_k), ..., s(a_{k-N_p})]^T$ is the signal vector and the pattern $\{a_k, ..., a_{k-N_p}\}$ is specified by the edge e_k.

If we plug the PDNP path metric (3.7) into (3.1), we get the following expression for the path metric margin:

$$\begin{aligned} M_k = \sum_{i=0}^{l-1} \{ & \ln v(\hat{\mathbf{a}}_{k-i}) + \frac{1}{v(\hat{\mathbf{a}}_{k-i})}([1,\ -\mathbf{p}^T(\hat{\mathbf{a}}_{k-i})](\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p})))^2 \\ & - \ln v(\mathbf{a}_{k-i}) - \frac{1}{v(\mathbf{a}_{k-i})}([1,\ -\mathbf{p}^T(\mathbf{a}_{k-i})](\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p})))^2 \}, \end{aligned} \tag{3.8}$$

where $\mathbf{p}(\mathbf{a})$ and $v(\mathbf{a})$ represent the coefficients of noise predictors and variance for pattern $\mathbf{a}$. Substituting (3.8) into (3.4) leads to the following AMBER update equations for $\mathbf{c}$, $\mathbf{s}(\,\cdot\,)$, $\mathbf{p}(\,\cdot\,)$ and $v(\,\cdot\,)$:

$$\begin{aligned}
\mathbf{c}_{k+1} &= \mathbf{c}_k + \lambda u(\tau - M_k) \sum_{i=0}^{l-1} \{ \frac{1}{v(\hat{\mathbf{a}}_{k-i})} \mathbf{p}'(\hat{\mathbf{a}}_{k-i}) (\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p})) \mathbf{R}'_{k-i} \mathbf{p}'(\hat{\mathbf{a}}_{k-i})^T \\
&\qquad - \frac{1}{v(\mathbf{a}_{k-i})} \mathbf{p}'(\mathbf{a}_{k-i}) (\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p})) \mathbf{R}'_{k-i} \mathbf{p}'(\mathbf{a}_{k-i})^T \}
\end{aligned}$$

for each $i \in \{0, 1, ..., l-1\}$

$$\mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p})_{k+1} = \mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p})_k + \lambda u(\tau - M_k) \frac{\mathbf{p}'(\mathbf{a}_{k-i})}{v(\mathbf{a}_{k-i})} (\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p})) \mathbf{p}'(\mathbf{a}_{k-i})^T$$

$$\mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p})_{k+1} = \mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p})_k - \lambda u(\tau - M_k) \frac{\mathbf{p}'(\hat{\mathbf{a}}_{k-i})}{v(\hat{\mathbf{a}}_{k-i})} (\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p})) \mathbf{p}'(\hat{\mathbf{a}}_{k-i})^T$$

$$\begin{aligned}
\mathbf{p}(\mathbf{a}_{k-i})_{k+1} = \mathbf{p}(\mathbf{a}_{k-i})_k + \lambda u(\tau - M_k) \frac{\mathbf{p}'(\mathbf{a}_{k-i})}{v(\mathbf{a}_{k-i})} (\mathbf{y}_{k-i}^{k-i-N_p} \\
- \mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p}))(\mathbf{y}_{k-i-1}^{k-i-N_p} - \mathbf{s}(\mathbf{a}_{k-i-1}^{k-i-N_p}))
\end{aligned}$$

$$\begin{aligned}
\mathbf{p}(\hat{\mathbf{a}}_{k-i})_{k+1} = \mathbf{p}(\hat{\mathbf{a}}_{k-i})_k - \lambda u(\tau - M_k) \frac{\mathbf{p}'(\hat{\mathbf{a}}_{k-i})}{v(\hat{\mathbf{a}}_{k-i})} (\mathbf{y}_{k-i}^{k-i-N_p} \\
- \mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p}))(\mathbf{y}_{k-i-1}^{k-i-N_p} - \mathbf{s}(\hat{\mathbf{a}}_{k-i-1}^{k-i-N_p}))
\end{aligned}$$

$$v(\mathbf{a}_{k-i})_{k+1} = v(\mathbf{a}_{k-i})_k - \lambda u(\tau - M_k)(\frac{1}{v(\mathbf{a}_{k-i})} - \frac{(\mathbf{p}'(\mathbf{a}_{k-i})(\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\mathbf{a}_{k-i}^{k-i-N_p})))^2}{v^2(\mathbf{a}_{k-i})})$$

$$v(\hat{\mathbf{a}}_{k-i})_{k+1} = v(\hat{\mathbf{a}}_{k-i})_k + \lambda u(\tau - M_k)(\frac{1}{v(\hat{\mathbf{a}}_{k-i})} - \frac{(\mathbf{p}'(\hat{\mathbf{a}}_{k-i})(\mathbf{y}_{k-i}^{k-i-N_p} - \mathbf{s}(\hat{\mathbf{a}}_{k-i}^{k-i-N_p})))^2}{v^2(\hat{\mathbf{a}}_{k-i})}), \tag{3.9}$$

where $\mathbf{p}'(\mathbf{a}) = [1, -\mathbf{p}^T(\mathbf{a})]$ and $\mathbf{R}'_{k-i} = [\mathbf{r}_{k-i}, ..., \mathbf{r}_{k-i-N_p}]$. The pseudocode of a PDNP Viterbi detector whose parameters are adapted according to the proposed AMBER algorithm is shown in Algorithm 2. Code lines 4 to 9 are the conventional Viterbi detector with PDNP branch metrics, while lines 10 to 19 implement the AMBER algorithm, which adapts the parameters of the PDNP branch metrics whenever the margin falls below the threshold.

3.2.3 AMBER for 2D-PDNP

Here we show how the AMBER algorithm can be applied to a multitrack detector that uses 2D-PDNP, where two readers spanning a pair of neighboring tracks are used to jointly detect the bits on the two tracks. Synchronous samples of the two readback waveforms are filtered by a two-input two-output equalizer with N_c coefficients (each a 2×2 matrix), represented by a $2 \times 2N_c$ matrix $\mathbf{C}$, resulting in the vector output y_k at time k. The equalizer outputs are then passed to a 2D-PDNP multitrack detector. The 2D bit pattern $\mathbf{A}$ is a matrix of bits with two rows, one for each track. Associated with each 2D bit pattern is a signal level vector $\mathbf{s}$, a standard deviation diagonal matrix $\mathbf{\Lambda}$, and a set of matrix-valued predictor

Algorithm 2 A PDNP Viterbi detector with parameters adapted by AMBER

Input: Equalizer outputs $\{y_k\}$; initial values s_0, $\mathbf{p}_0$ and v_0 for each pattern; step size λ; threshold τ; training bits $\{a_1, a_2, ..., a_L\}$; termination conditions.

Output: AMBER PDNP parameters s, $\mathbf{p}$ and v for each pattern.

repeat
  $\Phi_0(0) = 0,\ \Phi_0(p) = \infty\ \forall p \neq 0$
  for $k = 0$ to L **do**
    for $q = 0$ to $Q - 1$ **do**
      for $p \in$ predecessors(q) **do**
        $p^* = \text{argmin}_p\{\Phi_k(p) + \gamma_k(p, q)\}$
        $\Phi_{k+1}(q) = \Phi_k(p^*) + \gamma_k(p^*, q)$
        $\pi_{k+1}(q) = p^*$
      end for
      if q is the correct state at time k **then**
        Calculate M_k using (3.8)
        if $M_k < \tau$ **then**
          Trace back to get the correct bit subsequence $\{a_k, ..., a_{k-l+1}\}$
          Trace back to get the competing bit subsequence $\{\hat{a}_k, ..., \hat{a}_{k-l+1}\}$
          Get the separation length l
          $\lambda_H = \lambda w_H(\hat{\mathbf{a}}_k^{k-l+1} - \mathbf{a}_k^{k-l+1})$
          Update s_{k+1}, $\mathbf{p}_{k+1}$ and v_{k+1} for involved patterns using (3.9)
        end if
      end if
    end for
  end for
until Termination conditions are satisfied

coefficients $\{\mathbf{P}_0, \mathbf{P}_1, ..., \mathbf{P}_{N_p-1}\}$. The number of parameters is $2^b(5+4(N_p-1))+4N_c$ (for each of the 2^b patterns there are two signal levels, two residual variances, and $4(N_p-1)+1$ predictor coefficients), a number which can easily reach into the hundreds, depending on the number of bits b in each 2D pattern, the number N_p of matrix-valued predictor coefficients, and the number N_c of matrix-valued equalizer coefficients.

The branch metric for an edge e in the 2D-PDNP Viterbi detector is shown in (2.10). By plugging (2.10) into (3.1) and applying the AMBER algorithm, we arrive at the update equations for $\mathbf{C}$, $\mathbf{\Lambda}$, $\mathbf{s}$ and $\mathbf{P}_i$.

3.3 The Exponential Assumption and Hyper-Parameter Optimization

The threshold parameter τ of the AMBER algorithm is not arbitrary. Instead, it must be chosen carefully to ensure good performance and avoid trivial solutions with bad performance. In this section, we explore the role of τ and propose a strategy for its optimization.

3.3.1 Exponential Assumption

The basis for our analysis is the observation that the *tail* of the PDF for the path margin M_k often appears to have an exponential shape. For example, consider the upper-left inset of Fig. 3.2, which shows an experimentally measured PDF for the margin M_k in a single-track PDNP Viterbi detector operating on a quasi-micromagnetic simulated channel with a track pitch of 22.1 nm and a 70% centered reader. The bottom of Fig. 3.2 shows a close fit between the tail of the PDF and an exponential distribution (the red dashed curve).

Because we observed similar good fits to an exponential distribution over a wide range of channel conditions and detector parameters, we were encouraged to adopt the exponential model for the PDF tail described below, to facilitate analysis. It should be noted that the tail is not strictly speaking exponentially distributed, and that ultimately the value of the AMBER algorithm rests not on the exponential assumption that facilitates

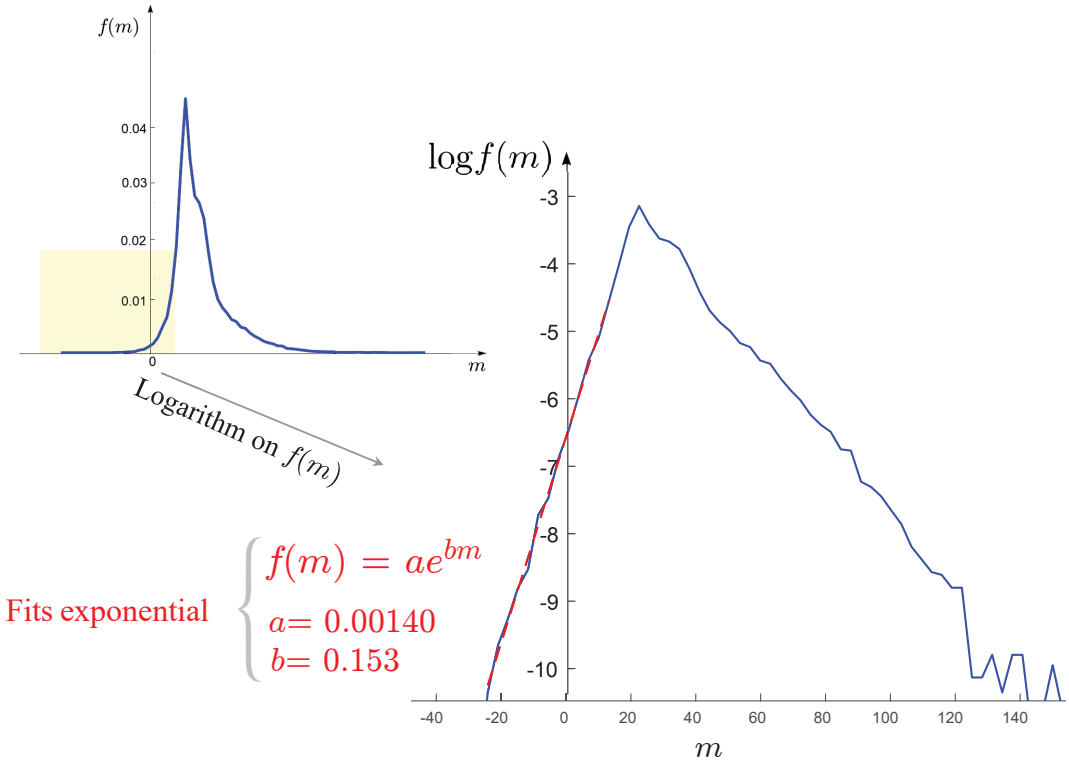


Figure 3.2: Margin PDF and its tail fit to an exponential function.

some of its analysis but on the good experimental results of the algorithm itself (see Section 3.4).

When the *tail* of the probability density function for M_k follows an exponential distribution:

$$f(m) = a(\mathbf{\Theta})e^{b(\mathbf{\Theta})m},\ m < \tau, \tag{3.10}$$

where a and b are parameters that depend on $\mathbf{\Theta}$, then the cost functions (3.2) and (3.3) reduce to:

$$J(\mathbf{\Theta}) = \frac{a(\mathbf{\Theta})}{b(\mathbf{\Theta})}, \tag{3.11}$$

$$J_\tau(\mathbf{\Theta}) = \frac{a(\mathbf{\Theta})}{b^2(\mathbf{\Theta})}e^{b(\mathbf{\Theta})\tau}. \tag{3.12}$$

Straightforward differentiation of (3.11) and (3.12) results in:

$$\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}) = \frac{1}{b^2(\boldsymbol{\Theta})}(\nabla a(\boldsymbol{\Theta}) b(\boldsymbol{\Theta}) - a(\boldsymbol{\Theta}) \nabla b(\boldsymbol{\Theta})), \tag{3.13}$$

$$\nabla_{\boldsymbol{\Theta}} J_\tau(\boldsymbol{\Theta}) = \frac{e^{b(\boldsymbol{\Theta})\tau}}{b^3(\boldsymbol{\Theta})}(\nabla a(\boldsymbol{\Theta}) b(\boldsymbol{\Theta}) - (2 - b(\boldsymbol{\Theta})\tau) a(\boldsymbol{\Theta}) \nabla b(\boldsymbol{\Theta})). \tag{3.14}$$

Let $\boldsymbol{\Theta}^*$ denote the set of parameters that minimizes $J(\boldsymbol{\Theta})$, so that substituting $\boldsymbol{\Theta}^*$ into (3.13) yields $\nabla_{\boldsymbol{\Theta}} J(\boldsymbol{\Theta}^*) = \mathbf{0}$. Let $\tau* = 1/b(\boldsymbol{\Theta}^*)$. Substituting $\tau = \tau*$ into (3.14) reveals that the same $\boldsymbol{\Theta}^*$ that minimizes $J(\boldsymbol{\Theta})$ also minimizes $J_{\tau*}(\boldsymbol{\Theta})$. The implication of this observation is that, when the margin tail is exponential, and when the AMBER threshold is chosen carefully (according to $\tau = 1/b(\boldsymbol{\Theta}^*)$), the cost function $J(\boldsymbol{\Theta})$ can be minimized by the AMBER algorithm.

3.3.2 Threshold

As stated there is a circular flaw: the optimal value for τ depends on the optimal value $\boldsymbol{\Theta}^*$ for the parameter set. Clearly there would be no need for AMBER or its threshold if $\boldsymbol{\Theta}^*$ were already known. To help break this cycle, we make use of the following fixed-point relationship:

Corollary 1. *Let $\boldsymbol{\Theta}(\tau)$ denote the parameter set that minimizes $J_\tau(\boldsymbol{\Theta})$. If τ satisfies the fixed-point relationship $\tau = \frac{1}{b(\boldsymbol{\Theta}(\tau))}$, then $\tau = \tau*$ and $\boldsymbol{\Theta}(\tau) = \boldsymbol{\Theta}^*$.*

Inspired by this corollary, we propose a fixed-point iterative strategy for automatically finding the best threshold. Starting with an arbitrary threshold, we run the AMBER algorithm until it converges, fit the margin tail that results to an exponential shape with parameter b, and then set the new threshold to $1/b$. This process is repeated until the threshold converges. The pseudocode of the proposed iterative algorithm is shown in Algorithm 3.

As an illustration of how Algorithm 3 works, consider the example of a conventional ten-coefficient equalizer followed by a single-track two-state Viterbi detector without

PDNP, so that the parameter set $\mathbf{\Theta}$ consists of ten equalizer coefficients and four signal levels. To ensure that the optimal $\mathbf{\Theta}$ is unique, we constrain both the equalizer energy and the signal level energy. The input to the detector is the Ehime waveforms described in Section 2.5.1, based on a 70% centered reader and a track pitch of 24.1 nm.

We sweep the threshold τ from 0 to 3, and for each value we use the AMBER algorithm to find the $\mathbf{\Theta}(\tau)$ that minimizes $J_\tau(\mathbf{\Theta})$. We then estimate the path metric margin PDF and fit its tail (for $M_k < \tau$) to an exponential shape, resulting in a b parameter we denote $b(\mathbf{\Theta}(\tau))$. In Fig. 3.3 we plot $1/b(\mathbf{\Theta}(\tau))$ versus τ in red (right-hand scale). The goal of the iterative algorithm is to find the value of τ where the fixed-point relationship is satisfied, namely where the dashed line (representing τ) intersects the red curve (representing $1/b(\mathbf{\Theta}(\tau))$). Starting with an arbitrary initial value of $\tau_0 = 2$, the algorithm 3 produces $\tau_1 = 1.1$, $\tau_2 = 0.75$, and $\tau_3 = 0.75$, converging quickly after only three iterations. The blue trajectory graphically illustrates how the algorithm bounces between the red and dashed curves until it converges to their intersection. Finally, overlaid on the same graph, we also plot $J(\mathbf{\Theta}(\tau))$ from (3.2) as a function of τ in black (left scale). Observe that the value of τ that minimizes this cost function coincides with the value of τ that satisfies the fixed-point relationship.

Algorithm 3 Iterative algorithm to ensure optimal τ

Input: Initial threshold τ_0, stop criterion ϵ
Output: Optimal τ and $\mathbf{\Theta}$.

1: $i = 0$
2: **repeat**
3: Run AMBER to get $\mathbf{\Theta}(\tau_i)$ that minimizes $J_{\tau_i}(\mathbf{\Theta})$
4: Fit steady-state margin tail PDF (for $m < \tau$) to an exponential distribution, and estimate its $b(\mathbf{\Theta}(\tau_i))$
5: Set $\tau_{i+1} = \frac{1}{b(\mathbf{\Theta}(\tau_i))}$
6: $i = i + 1$
7: **until** $|\tau_i - \tau_{i-1}| < \epsilon$

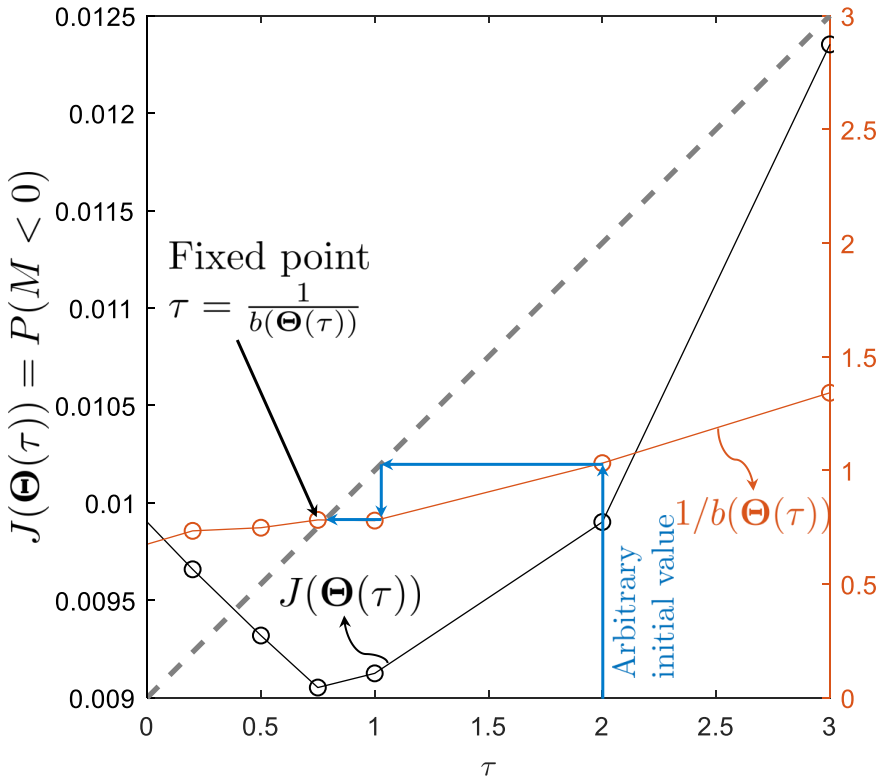


Figure 3.3: The fixed-point relationship between τ and $1/b(\boldsymbol{\Theta}(\tau))$. The simulation is run in training mode.

3.3.3 Hamming-Weighted Step Size

As written, when the margin tail is exponential and the threshold τ is optimized, the AMBER algorithm of (3.4) minimizes $J(\boldsymbol{\Theta}) = P(M_k < 0)$. If each instance of the error event $M_k < 0$ led to a single bit error, then this would be equivalent to minimizing BER. However, because some error events cause more bit errors than others, the AMBER algorithm as written does not minimize BER. We can improve the BER performance of the AMBER algorithm by a simple modification of the step size λ in (3.4), namely, by introducing an adaptive Hamming-weighted step size $\lambda = \lambda_0 w_H$, where λ_0 is a fixed nominal step size, and w_H is shorthand for the number of message bits that differ between the correct subpath $\{\theta_{k-\ell_k}, \ \ldots \ \theta_k\}$ and the competing subpath $\{\hat{\theta}_{k-\ell_k}, \ \ldots \ \hat{\theta}_k\}$. The Hamming-weight factor ensures that a large burst of bit errors will cause a bigger change

in parameters than a small burst of errors.

To illustrate the benefit of the Hamming weight factor, consider again the example from the previous section (a ten-coefficient equalizer followed by a two-state Viterbi detector without PDNP, based on a 70% centered reader and a track pitch of 24.1 nm). Let us fix the equalizer to be MMSE, and further constrain the signal levels to be symmetric: either $\pm s_1$ when two consecutive bits are the same, or $\pm s_2$ when two consecutive bits are different. There are thus only two parameters to optimize: $\mathbf{\Theta} = \{s_1, s_2\}$.

In Fig. 3.4 we show a contour plot of BER as a function of the parameters s_1 and s_2. Overlaid on the same graph is a pair of trajectories for the AMBER algorithm, both starting from the same arbitrary initial condition $(-1.1,\ 0.9)$: The red curve has no Hamming-weight factor, while the blue curve includes the Hamming-weight factor. In both cases the initial λ_0 is 10^{-4}, and it decays with a half-life of 25. After convergence the BER with the Hamming factor is 0.0130, while the BER without the Hamming factor is 0.0133. Also shown in the figure is the MBER point, found by exhaustive search, which achieves BER $= 0.0120$, along with the MMSE point, which achieves BER $= 0.0164$. This example illustrates not only the benefit of the Hamming weight factor in the step size, but also the general suboptimality of MMSE with respect to BER, and further the effectiveness of the AMBER algorithm to seek out the minimum of the BER surface.

3.4 Quantitative Results

We test our algorithm on Ehime waveforms. To better demonstrate the performance of the AMBER algorithm, no electronic noise is added.

3.4.1 Performance of AMBER PDNP Detection

We use the second track for training parameters, and the third (middle) track to detect in the following simulation. We now present numerical results for the AMBER algorithm presented in Section 3.2, which applies to the case of a fixed MMSE equalizer followed by

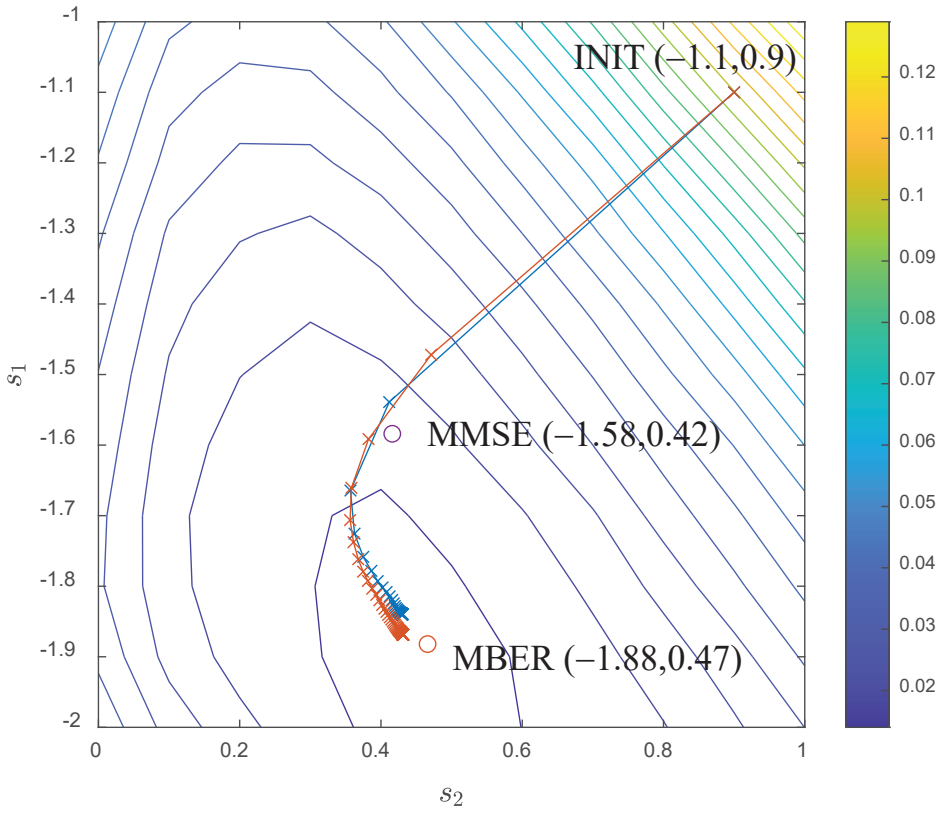


Figure 3.4: BER contour with trajectories of AMBER parameters w/o Hamming-weight factor.

a PDNP Viterbi detector. The parameters to be adapted are the parameters of the PDNP detector, namely the pattern-dependent signal levels, noise variances, and predictor coefficients. We consider two-bit patterns, so that the number of patterns is four, and a single predictor coefficient for each pattern. The total number of parameters being adapted is thus twelve: four signal levels, four variances, and four predictor coefficients.

To start with, we investigate the convergence properties by plotting BER versus iterations in Fig. 3.5. The AMBER algorithm is tested on waveforms of track pitch 24.1 nm with a 70% centered reader. A decaying step size 10^{-4} with a half-life of 100 iterations is employed, and the parameters are initialized to those of a non-PDNP Viterbi detector. We train the parameters on track two by the AMBER algorithm and plot the training BER curve in blue versus iterations. The red curve is the testing BER achieved by detecting track three with the training parameters after each iteration. The figure shows

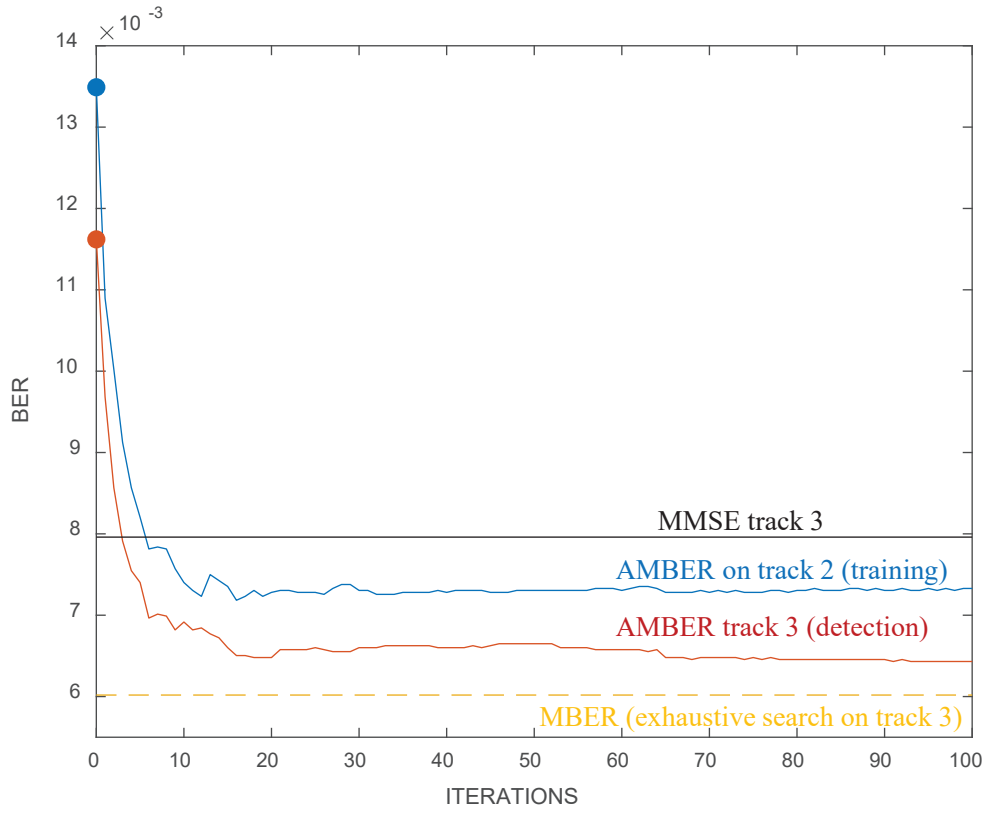


Figure 3.5: Learning and testing curves for the AMBER PDNP algorithm.

that the AMBER algorithm converges fast and the BER gets close to the MBER line found via an exhaustive search on track three, and becomes stable after 60 iterations, achieving a 20% reduction in BER compared to MMSE line plotted in black. We also plot the parameter evolution in Fig. 3.6. The signal levels (blue dashed curves), variances (red solid curves) and predictor coefficients (black dot curves) converge fast.

3.4.2 Performance of AMBER 2D-PDNP Detection

In this simulation, we consider a scenario where two readers detect two tracks. We use tracks two and three to train the parameters, and we use the resulting parameters to detect the bits written on track three and four, with eleven matrix-valued equalizer coefficients. The detector has sixteen 2D patterns, and there are only two prediction matrices $\mathbf{P}_0$ and $\mathbf{P}_1$ ($N_p = 1$), so that the number of states with 2D-PDNP is sixteen. The total number of

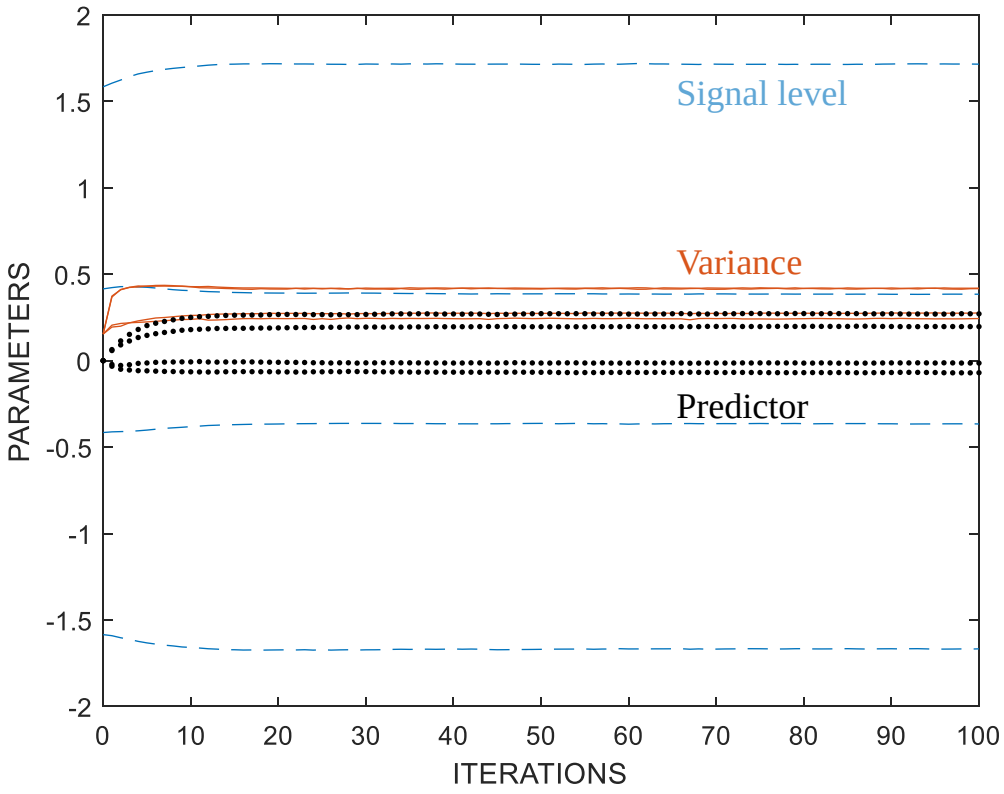


Figure 3.6: Parameter evolution for AMBER PDNP.

parameters is 188. We do not consider longer pattern lengths because there are not sufficient training data for each pattern. MMSE parameters and non-PDNP Viterbi parameters can be used as initials for the AMBER algorithm. The residual variances for non-PDNP Viterbi are the same for each pattern and can be arbitrarily chosen since they will not affect the BER. Partly because of this, we find non-PDNP Viterbi initials usually converge to a lower BER.

We investigate the convergence properties of the AMBER algorithm by plotting BER (averaged over the two tracks) versus iterations. In this experiment, there are two 130% readers one-eighth track offset inside the two tracks of interest, and the track pitch is 26.1 nm. The step size is initialized to 10^{-5} and decreases exponentially with a half-life of 300 iterations, and the threshold is $\tau = 1$. Fig. 3.7 shows the training curve for the AMBER algorithm (in blue), where we see its convergence after about 450 iterations. The detection

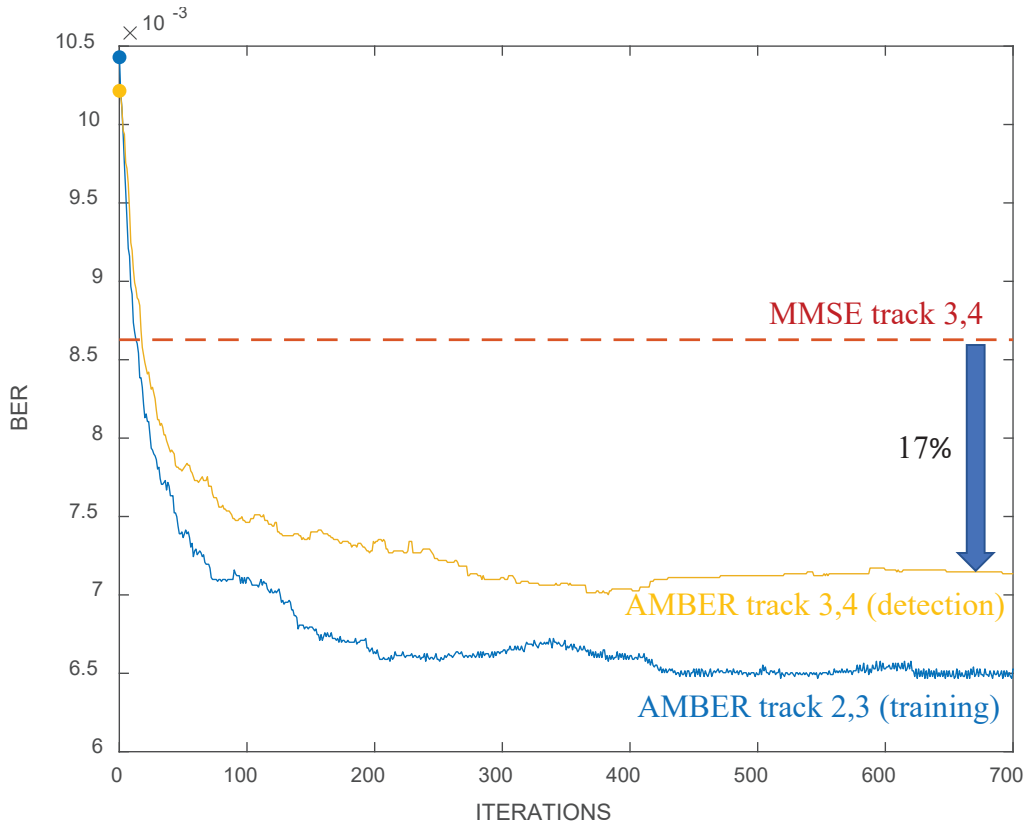


Figure 3.7: Learning and testing curves for the AMBER 2D-PDNP algorithm.

curve converges 50 iterations faster. Compared with 2D-MMSE parameters, 2D-AMBER parameters achieve a 17% decrease in BER.

3.4.3 Optimize the Parameter Space

The number of parameters of 2D-PDNP grows exponentially as the pattern bits increase. While in principle we could adapt all of them, it may not be necessary or desirable. We can optimize the parameters space by removing the redundant and negligible parameters.

We divide all parameters to be adapted into three categories: equalizer coefficients, signal levels and PDNP parameters (including residual variances and predictor coefficients). We can apply the AMBER algorithm to part of the parameters and keep the rest MMSE parameters. Let "1" and "0" represent turning on and off the AMBER algorithm. We use three binary digits to represent the described categories. For example,

Table 3.1: BER and number of parameters adapted for all parameter schemes.

	BER ($\times 10^{-3}$)	#Parameters to be adapted
111	7.00	188
011	7.09	144
100	7.35	44
110	7.37	76
101	7.39	156
001	7.94	112
010	8.24	32
000	8.63	0

"101" means the scheme of AMBER equalizer coefficients, MMSE signal levels and AMBER PDNP parameters.

We continue to use the same waveform settings in Section 3.4.2. We test and list the BER and the number of parameters adapted for each scheme in Table 3.1 in ascending order by BER. According to the BER, we rank all schemes into three tiers. Tier 1: "111" and "011". Tier 2: "100", "110", and "101". Tier 3: "001", "010", and "000". "111" and "011" have comparable performance but "011" needs fewer parameters. "100" outperforms the others among tier 2 because of fewer parameters. If a system requires less complexity and a relatively good BER, then "100" is a perfect choice. We do not recommend the schemes in tier 3 in any cases.

We can draw the following conclusions from this experiment. It is redundant to adapt all parameters. Fixing MMSE equalizer coefficients and adapting the rest parameters can reach the optimal BER area of parameter space, compared with the performance of "111" and "011". However, only adapting the linear equalizer coefficients has limited ability to reduce BER when signal levels and PDNP parameters are fixed according to the performance of "011" and "100". We also notice that equalizer coefficients may "fight" with the others. The roles of the equalizer and predictors have significant overlap, since both yield an overall filtering operation on the ADC samples, and we find that when both are adapted they may interfere with each other and perform worse than only one is adapted. Signal levels and

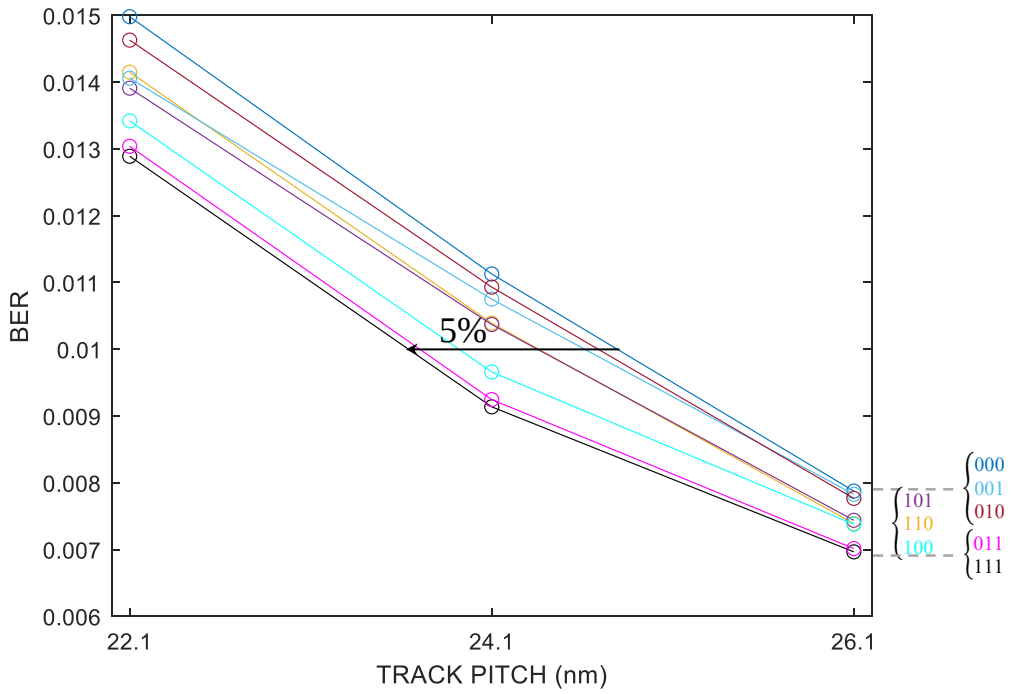


Figure 3.8: BER performance of all schemes.

PDNP parameters need to be adapted together to reduce the BER efficiently according to the performance of "011", "001", "010" and "111", "101", "110". However, the BER ordering may vary under different waveform settings.

We also test all schemes at track pitches of 22.1 nm, 24.1 nm and 26.1 nm with 70%, 85% and 100% reader widths, respectively in Fig. 3.8. One reader is placed at the position of -7/8 and the other is placed at the position of -1/8 as labeled in Fig. 2.4. As we can see, "011" and "111" have a similar performance better than others. "100" shows a similar BER with "101 and "110" at 26.1 nm but better than them at 22.1 nm and 24.1 nm. At the BER of 10^{-2}, "111" AMFER parameters achieve 5% areal density gain over "000" MMSE parameters.

In Fig. 3.9, we explore the performance and complexity trade-off. The x-axis shows the complexity measured by the number of parameters while the y-axis shows the grains per bit required at the BER of 10^{-2}. A better scheme needs a narrower track pitch given

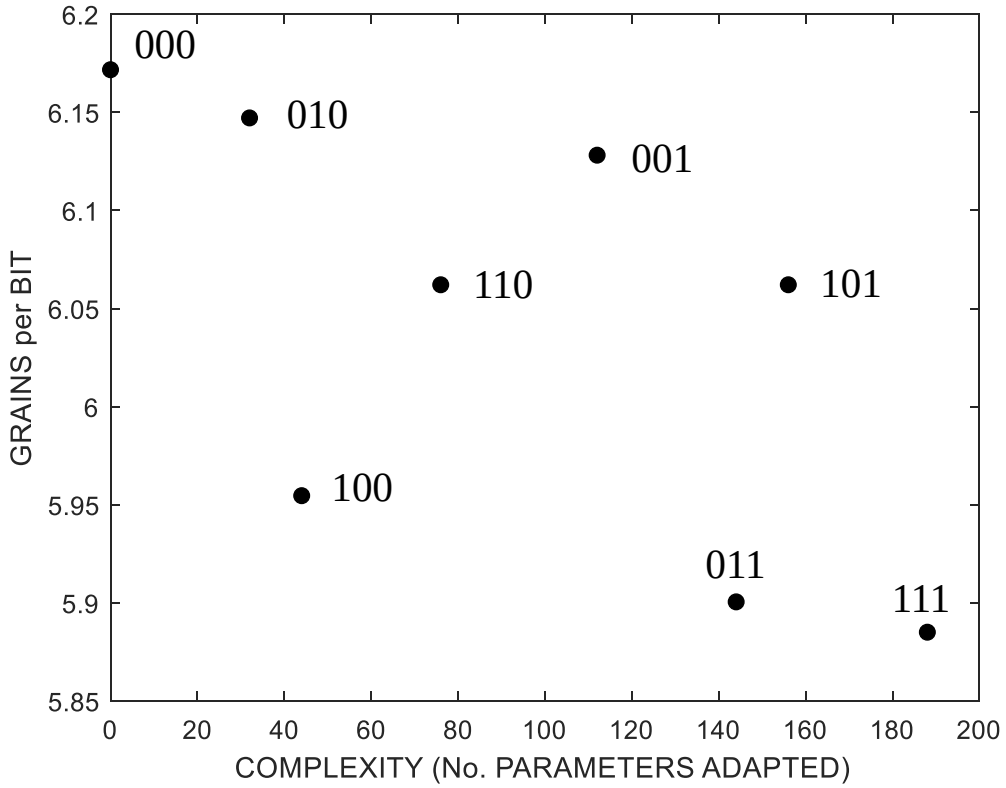


Figure 3.9: Performance and complexity trade-off of different schemes.

a fixed BER, in turn, leading to a higher areal density which can be represented as grains per bit. From the figure, "111" is the best performance-driven scheme, while "100" is a performance-complexity balanced scheme.

3.5 Summary

In this chapter, we present an adaptive algorithm for tuning the parameters of a PDNP detector so as to minimize BER. The AMBER algorithm updates the parameters whenever the path metric margin between the competing path and the correct path in the trellis-based detector is smaller than a threshold; the parameters are then updated so as to minimize the proposed cost function. This algorithm applies for all the trellis-based detectors and we derive the update equations for non-PDNP Viterbi, 1D-PDNP and 2D-PDNP detectors. We

also provide a fixed-point iterative strategy for optimizing the AMBER threshold. To deal with the number of parameters exploding problem, we propose the method to adapt part of parameters and keep the rest MMSE parameters. We also explore the performance and complexity trade-off for all the schemes.

Results on simulated waveforms demonstrate that the AMBER 2D-PDNP parameters measurably outperform conventional MMSE parameters. We expect further gains in performance by increasing the length of the training sequences, pattern lengths, and the number of prediction coefficients.

CHAPTER 4

MINIMUM-FRAME-ERROR TUNING FOR TURBO DETECTION

4.1 Introduction

In Chapter. 2, the strategy for choosing the read channel parameters is based on the MMSE criterion. In Chapter. 3, we provide the AMBER algorithm to optimize the parameters aiming to minimize BER. The BER after detection, while more relevant than MSE, is still not as relevant to the end user as the frame-error rate (FER) after error-control decoding, because the FER is what ultimately determines areal density. In a state-of-the-art read channel, a soft-output detector and a soft-input error-control decoder work iteratively in a turbo fashion with soft information transferring between them. The soft-output detector is typically either the BCJR or SOVA [59].

This chapter aims to define a criterion that quantifies the "quality" of the soft output and devise an algorithm to choose parameters that can maximize the soft output "quality" of the detectors for both single-track and multitrack systems.

This chapter is organized as follows. In Section 4.2, we review EXIT charts. In Section 4.3, we propose the minimum-FER tuning strategy and the adaptive minimum-frame-error rate algorithm. In Section 4.4, we evaluate the performance of the proposed algorithm, and compare it to traditional MMSE parameters. In Section 4.5, we summarize this chapter.

4.2 The EXIT Chart

We consider the magnetic recording read channel illustrated in Fig. 4.1[52]. A vector $\mathbf{r}$ of readback waveform samples is fed to an equalizer, producing a vector $\mathbf{y}$ of equalizer outputs. An APP detector has two inputs: the equalizer output $\mathbf{y}$, and a vector $\boldsymbol{\lambda}_2 =$

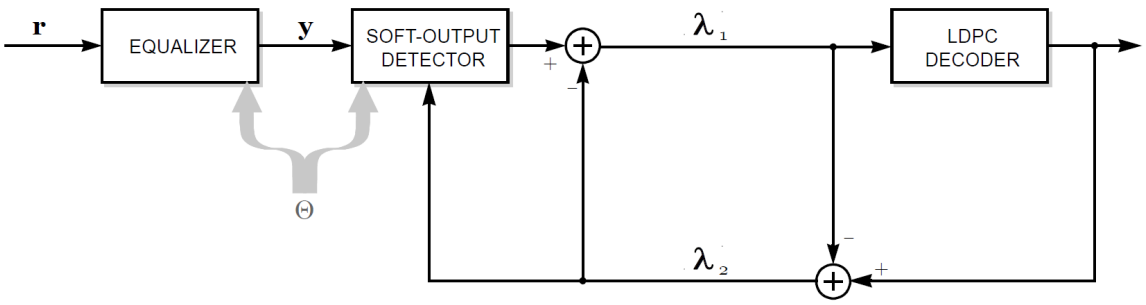


Figure 4.1: A soft-output detector interacting with an LDPC decoder in an iterative (turbo) fashion.

$[\lambda_{2,0}, ..., \lambda_{2,N-1}]^T$ of *a priori* LLR's about the N written coded bits $\mathbf{a} = [a_0, ..., a_{N-1}]^T$, where $\lambda_{2,k} = \ln(\frac{P(a_k=1)}{P(a_k=0)})$. Based on these inputs, the APP detector estimates the vector $\mathbf{L} = [L_0, ..., L_{N-1}]^T$ of APP LLR's about the written coded bits, where $L_k = \ln(\frac{P(a_k=1|\mathbf{y})}{P(a_k=0|\mathbf{y})})$. The difference $\boldsymbol{\lambda}_1 = \mathbf{L} - \boldsymbol{\lambda}_2$ is a vector of *extrinsic* LLR's, which are fed as a priori information to the LDPC decoder. In a symmetric fashion, the LDPC decoder estimates the APP LLR's about the coded bits, which after subtracting the decoder a priori information leads to the vector $\boldsymbol{\lambda}_2$ of extrinsic information from the decoder. As shown in the figure, $\boldsymbol{\lambda}_2$ is then interpreted as a priori information by the detector in the next iteration. The detector and decoder thus work in an iterative (turbo) fashion until a coded frame is successfully decoded or a maximum iteration number is reached.

As indicated in Fig. 4.1, we use $\boldsymbol{\Theta}$ to denote all of the parameters to be optimized, including the equalizer coefficients and any parameters within the detector. For simplicity, Fig. 4.1 omits any interleavers that might be used to prevent burst errors.

A Gaussian distribution is characterized by two parameters, mean and variance. A Gaussian distribution is said to be *consistent* when the variance is twice the mean, and is thus characterized by a single parameter. Likewise, the *a priori* information $\boldsymbol{\lambda}_2$ is said to be *consistent* when the random variables $\{z_{2,k} = \lambda_{2,k} a_k\}$ are i.i.d. $\mathcal{N}(\mu_2, 2\mu_2)$ for some parameter μ_2. These random variables will be neither consistent nor independent in general; nevertheless (as recognized in [48]) they can be roughly approximated as such, and we will see that optimizing the detector is dramatically simplified when the a priori

information coming from the decoder is assumed to be consistent.

The quality of the a priori LLR's is captured by the mutual information $I_2 \triangleq I(\{\lambda_{2,k}\}; \{a_k\})$ between the written bits and the a priori LLR's [48], which (under the consistent Gaussian approximation) reduces to (see the Appendix B):

$$I_2 = 1 - E(\log_2(1 + e^{-Z_2})), \tag{4.1}$$

where $Z_2 \sim \mathcal{N}(\mu_2, 2\mu_2)$.

The mutual information $I_1 \triangleq I(\{\lambda_{1,k}\}; \{a_k\})$ after the detector will depend in a predictable way on the mutual information I_2 after decoding. Let

$$I_1 = T_1(I_2) \tag{4.2}$$

denote the function that relates I_2 to I_1; it is an increasing function that approaches $T_1(I_2) \to 1$ as $I_2 \to 1$. Similarly, let T_2 denote the function that relates I_1 to I_2:

$$I_2 = T_2(I_1). \tag{4.3}$$

The $T_1(\,\cdot\,)$ and $T_2(\,\cdot\,)$ functions are the so-called *extrinsic information transfer (EXIT)* functions. Given readback waveforms, the shape of T_1 is determined by the parameters of the equalizer and detector. These transfer functions must be measured empirically because closed-form expressions for them are unknown. We simplify the calculation of T_2 by implementing a time-averaged version of (4.1), which strictly speaking only applies when $\{a_k\lambda_{2,k}\}$ are i.i.d. consistent Gaussian, even when they are not:

$$I_2 = 1 - \frac{1}{N}\sum_{k=0}^{N-1} \log_2(1 + e^{-a_k\lambda_{2,k}}). \tag{4.4}$$

Likewise, the mutual information between the extrinsic LLR's after the detector and the

written bits can be estimated using:

$$I_1 = 1 - \frac{1}{N}\sum_{k=0}^{N-1} \log_2(1 + e^{-a_k \lambda_{1,k}}). \tag{4.5}$$

A chart showing both T_1 and T_2 is known as an EXIT chart, and is a powerful tool for understanding convergence of iterative detectors. We illustrate an EXIT chart with an example.

Example 1. Consider an ISI channel $H(z) = 0.5 + z^{-1} + 0.5z^{-2}$ with AWGN, and suppose the written bits are coded by a rate-$8/9$ LDPC code of length 16200 from DVB-S2 [60]. A five-tap equalizer and a two-tap partial-response monic target are jointly chosen according to the MMSE criterion, followed by a two-state SOVA detector. In Fig. 4.2 we plot the transfer function $T_1(I_2)$ versus I_2 for $\mathrm{SNR} = \sum_k h_k^2/(2\sigma_n^2)$ values of 6.7 dB, 6.3 dB, and 5.9 dB. T_1 curve is found by sweeping through all possible I_2 values, or equivalently through all possible μ_2 values, and for each one generate an artificial consistent Gaussian a priori vector $\boldsymbol{\lambda}_2 = \mu_2 \mathbf{a} + \mathbf{w}_2$, where the components of $\mathbf{w}_2$ are i.i.d. $\mathcal{N}(0, 2\mu_2)$. After SOVA uses $\boldsymbol{\lambda}_2$ as a priori information to generate $\mathbf{L}$, the extrinsic information $\boldsymbol{\lambda}_1 = \mathbf{L} - \boldsymbol{\lambda}_2$ is used to estimate I_1 via (4.5). Improved estimates are found by averaging these I_1 values over repeated trials. Also shown in the figure is the transfer curve $T_2^{-1}(I_2)$ for the LDPC decoder, which is independent of SNR and is found in a symmetric way. For each possible μ_1 generate a consistent Gaussian a priori vector $\boldsymbol{\lambda}_1 = \mu_1 \mathbf{a} + \mathbf{w}_1$, where the components of $\mathbf{w}_1$ are i.i.d. $\mathcal{N}(0, 2\mu_1)$. The extrinsic information $\boldsymbol{\lambda}_2$ is achieved by subtracting a priori information $\boldsymbol{\lambda}_1$ from the decoder outputs and is used to estimate I_2 via (4.4). As shown in the figure, the T_1 curve drops lower as the SNR decreases, and the two curves intersect when $\mathrm{SNR} = 5.9$ dB. The green dashed staircase curve shows a sample decoding trajectory for the case when $\mathrm{SNR} = 6.7$ dB. The trajectory starts at $(0,\ 0)$, bounces between the two transfer functions, and stops near $(1,\ 1)$, after the decoding is successful.

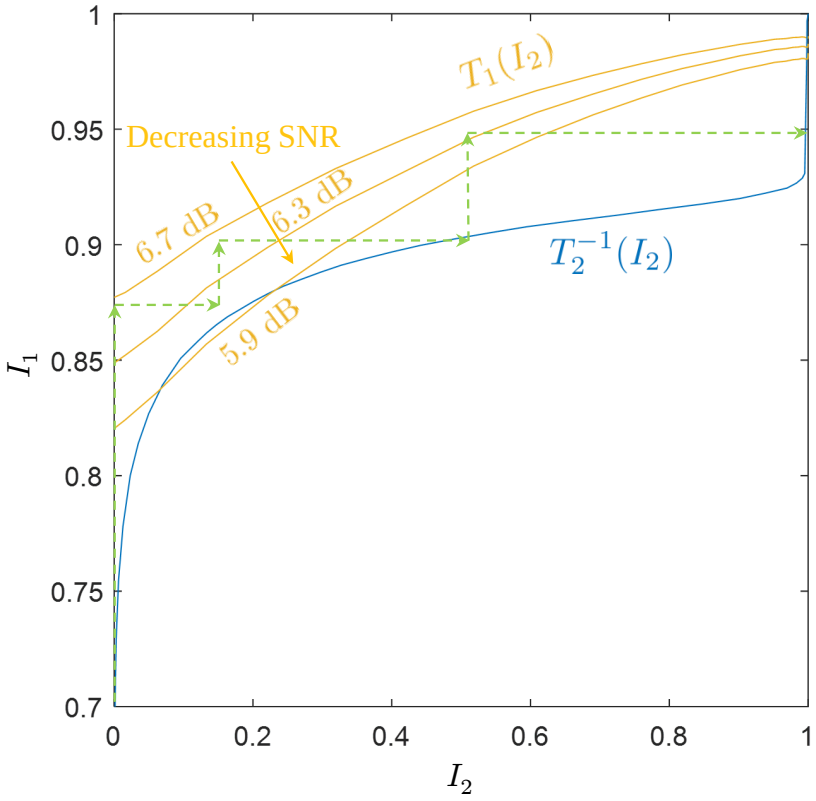


Figure 4.2: EXIT charts of Example 1 under different SNR's.

4.3 Adaptive Minimum-Frame-Error Rate Algorithm

Direct optimization of the parameters Θ to minimize FER would require an analytical expression for FER in terms of the parameters to be optimized; such an expression is unknown and likely to be unwieldy even if it were known. Instead, we propose to indirectly optimize FER by optimizing the EXIT chart. It has been shown in [48] that when the curves in an EXIT chart intersect, there is a high probability of decoding failure. In contrast, a clear tunnel between the two curves means the decoding is very likely to be successful. The wider the tunnel is, the faster the convergence will be.

Decoding success is not so much dependent on the entire shape of the two curves in an EXIT chart, but is instead determined largely by what the two curves look like when they

are closest to each other. Motivated by this observation, and with the understanding that a larger gap between the two curves leads to a higher probability of decoding successfully and faster decoding, we propose to choose the parameters $\mathbf{\Theta}$ so as to maximize the gap between the two curves at a particular value I_2^* for I_2, where I_2^* is chosen in the "bottleneck" region of the chart where the two curves are closest. In the example of Fig. 4.2, a value of I_2^* around 0.2 would be an appropriate choice when $\mathrm{SNR} = 5.9$ dB.

Since only the T_1 curve depends on $\mathbf{\Theta}$, maximizing the gap is equivalent to maximizing T_1. Exploiting (4.5), we propose to choose $\mathbf{\Theta}$ to minimize the following cost function:

$$J(\mathbf{\Theta}) = \frac{1}{N}\sum_{k=0}^{N-1} \log_2(1 + e^{-a_k \lambda_{1,k}^*}), \tag{4.6}$$

where $\{\lambda_{1,k}^*\}$ are the extrinsic LLR's produced by the detector when it is fed with a consistent Gaussian a priori vector $\boldsymbol{\lambda}_2 = \mu_2^* \mathbf{a} + \mathbf{w}_2$, where the components of $\mathbf{w}_2$ are i.i.d. $\mathcal{N}(0, 2\mu_2^*)$, where μ_2^* is the value of μ_2 that leads to I_2^* in (4.1).

As a result, the cost function does not depend on any features of the decoder or of the code itself. The proposed cost function requires knowledge of the written bits, which can be viewed as a form of training.

By either enlarging the bottleneck or opening a tunnel in the EXIT chart, we enhance the probability that the decoding trajectory passes through the bottleneck region. In the end, the FER will decrease. We should point out that maximizing T_1 at one point I_2^* may cause the mutual information to degrade at other values of I_2; if this effect is undesirable one could swap different parameter sets for different I_2 values, so that the detector parameters would change from one iteration to the next.

Applying the stochastic gradient algorithm to $J(\mathbf{\Theta})$ leads to the adaptive minimum-frame-error rate (AMFER) algorithm for adapting the parameters $\mathbf{\Theta}$:

$$\mathbf{\Theta}_{k+1} = \mathbf{\Theta}_k + \frac{\alpha a_k}{1 + e^{a_k \lambda_{1,k}^*}} \nabla_{\mathbf{\Theta}} \lambda_{1,k}^*, \tag{4.7}$$

where α is the step size. Note the factor $\alpha a_k/(1+e^{a_k\lambda^*_{1,k}})$ in (4.7) can be interpreted as a time-varying step size that is negligible when $a_k\lambda^*_{1,k}$ is large; only when $a_k\lambda^*_{1,k}$ is negative or small will the parameters change significantly.

4.3.1 AMFER for Single Track Detection

The AMFER algorithm of (4.7) is general and can be applied to a wide range of detector and equalizer architectures. Here we present a concrete example based on a 2^μ-state SOVA detector, where the parameters $\mathbf{\Theta}$ to be optimized are the equalizer coefficients $\mathbf{c}$, the noise standard deviation σ, and the signal levels $s(\mathbf{a}_k)$ associated with each bit pattern $\mathbf{a}_k = [a_k, ..., a_{k-\mu}]^T$. SOVA estimates the extrinsic LLR for a_k using:

$$\lambda^*_{1,k} = \hat{a}_k\Delta_k - \lambda_{2,k}, \tag{4.8}$$

where $\hat{a}_k$ is the bit chosen by the detector, Δ_k is chosen from a path metric margin M_k after time k (see Algorithm 4), and where the path metric margin M_k is the difference between the survivor path metric and the competing path metric:

$$\begin{aligned} M_k &= \sum_{i=0}^{\ell_k-1} -\log P(\bar{a}_{k-i}) + \frac{(\mathbf{c}^T\mathbf{r}_{k-i} - s(\bar{\mathbf{a}}_{k-i}))^2}{2\sigma^2} \\ &\qquad - \sum_{i=0}^{\ell_k-1}(-\log P(\hat{a}_{k-i}) + \frac{(\mathbf{c}^T\mathbf{r}_{k-i} - s(\hat{\mathbf{a}}_{k-i}))^2}{2\sigma^2}) \\ &= \sum_{i=0}^{\ell_k-1} \frac{\hat{a}_{k-i} - \bar{a}_{k-i}}{2}\lambda_{2,k-i} + \frac{1}{2\sigma^2}(2\mathbf{c}^T\mathbf{r}_{k-i}(s(\hat{\mathbf{a}}_{k-i}) \\ &\qquad - s(\bar{\mathbf{a}}_{k-i})) + s^2(\bar{\mathbf{a}}_{k-i}) - s^2(\hat{\mathbf{a}}_{k-i})), \end{aligned} \tag{4.9}$$

where ℓ_k is the separation length of the two paths, and $\bar{a}_k$ is the competing bit. Plugging (4.9) and (4.8) into the AMFER Algorithm (4.7), along with the relationship between M_k and Δ_k described in Algorithm (4.7), leads to the following update equations for the

parameters $\mathbf{\Theta}$:

$$\mathbf{c}_{k+1} = \mathbf{c}_k + \frac{a_k \hat{a}_k}{1 + e^{a_k \lambda^*_{1,k}}} \sum_{i=0}^{\ell_k - 1} \frac{\mathbf{r}_{k-i}}{\sigma^2} (s(\hat{\mathbf{a}}_{k-i}) - s(\bar{\mathbf{a}}_{k-i})),$$

$$\sigma_{k+1} = \sigma_k - \frac{a_k \hat{a}_k}{1 + e^{a_k \lambda^*_{1,k}}} \sum_{i=0}^{\ell_k - 1} \frac{1}{\sigma^3} (2\mathbf{c}^T \mathbf{r}_{k-i} (s(\hat{\mathbf{a}}_{k-i}) - s(\bar{\mathbf{a}}_{k-i})) + s^2(\bar{\mathbf{a}}_{k-i}) - s^2(\hat{\mathbf{a}}_{k-i})),$$

$$\text{for each } i \in \{0, 1, ..., \ell_k - 1\} :$$

$$s(\hat{\mathbf{a}}_{k-i})_{k+1} = s(\hat{\mathbf{a}}_{k-i})_k + \frac{a_k \hat{a}_k}{1 + e^{a_k \lambda^*_{1,k}}} (\frac{\mathbf{c}^T \mathbf{r}_{k-i} - s(\hat{\mathbf{a}}_{k-i})}{\sigma^2})$$

$$s(\bar{\mathbf{a}}_{k-i})_{k+1} = s(\bar{\mathbf{a}}_{k-i})_k - \frac{a_k \hat{a}_k}{1 + e^{a_k \lambda^*_{1,k}}} (\frac{\mathbf{c}^T \mathbf{r}_{k-i} - s(\bar{\mathbf{a}}_{k-i})}{\sigma^2}). \quad (4.10)$$

A common modification in iterative detectors (see [56]) is to introduce an extra parameter that scales the LLR's after the detector and before the decoder. Instead of choosing this parameter based on an ad hoc search based on empirical results, as is commonly done, it can be folded into $\mathbf{\Theta}$ and optimized by AMFER.

The pseudocode of a SOVA detector whose parameters are adapted according to the proposed AMFER algorithm is shown in Algorithm 4. Code lines 5 to 18 are the conventional SOVA detector, while lines 19 to 20 implement the AMFER algorithm.

4.3.2 AMFER for Multitrack Detection

In the multitrack detection system, the equalizer is replaced with a MIMO equalizer and the detector is replaced with a joint soft-output detector. We focus the case of detecting two tracks and generalizing more than tracks is straightforward. We implement crosstrack coding in the following way: a coded frame with a length N is cut into two halves. One is written on the first track and the other is on the second track. The benefit of crosstrack coding over single-track coding is that bit errors from one track can be corrected by information from the other track.

To embed the joint detector into turbo detection as shown in Fig. 4.1. We denote the first half $\{\lambda_{1,0}, ..., \lambda_{1,\frac{N}{2}-1}\}$ in $\boldsymbol{\lambda}_1$ is the APP LLR's from the first track after detection, and the second half $\{\lambda_{1,\frac{N}{2}}, ..., \lambda_{1,N-1}\}$ is the APP LLR's from the second track. After decoding, the first half of $\boldsymbol{\lambda}_2$ is the a priori information for the first track and the second half is the a priori information for the second track in the next detection. We can still use (4.7) to update

Algorithm 4 A SOVA detector with parameters adapted by AMFER

Input: Equalizer input $\mathbf{y}$; I_2^*; initial values $s(\ \cdot\ )_0$, $\mathbf{c}_0$ and σ_0; step size α; training bits $\{a_0, a_2, ..., a_{N-1}\}$; termination conditions.

Output: AMFER parameters $s(\ \cdot\ )$, $\mathbf{c}$ and σ.

Get μ_2^* from I_2^* by inverting (4.1)
repeat
  generate $\boldsymbol{\lambda}_2 = \mu_2^* \mathbf{a} + \mathbf{w}_2$,
  where the components of $\mathbf{w}_2$ are i.i.d. $\mathcal{N}(0, 2\mu_2^*)$
  Run Viterbi algorithm to get $\{\hat{a}_k\}$
  $\Phi_0(0) = 0,\ \Phi_0(p) = \infty\ \forall p \neq 0$
  for $k = 1$ to L **do**
    for $q = 0$ to $Q - 1$ **do**
      for $p \in$ predecessors(q) **do**
        $p^* = \text{argmin}_p\{\Phi_k(p) + \gamma_k(p, q)\}$
        $\Phi_{k+1}(q) = \Phi_k(p^*) + \gamma_k(p^*, q)$
        $\pi_{k+1}(q) = p^*$
        if q is the survivor state at time k **then**
          Calculate M_k using (4.9)
          $\Delta_k = \infty$
          Trace back to get the competing
          bit subsequence $\{\bar{a}_k, ..., \bar{a}_{k-\ell_k+1}\}$
          for $i = 1$ to $\ell_k - 1$ **do**
            if $\hat{a}_{k-i} \neq \bar{a}_{k-i}$ and $\Delta_{k-i} > M_k$ **then**
              $\Delta_{k-i} = M_k$
              Restore previous update on
              parameters at stage $k - i$, if any
              Update $\boldsymbol{\Theta}_k$ using (4.10)
            end if
          end for
        end if
      end for
    end for
  end for
until Termination conditions are satisfied

parameters. The detailed update equations for 2D-PDNP SOVA is a direct extension of 1D-PDNP SOVA depicted in Section 4.3.1.

4.4 Quantitative Results

4.4.1 Linear ISI Example

In this subsection we continue the linear ISI channel of Example 1. We set the maximum number of decoder iterations to ten, and the maximum number of overall (turbo) iterations to twenty. For the training process, we train the parameters (the equalizer, signal levels and residual noise variance) with one frame of 16200 bits for 500 epochs. MMSE initials are used for the first epoch. In each epoch, we generate different a priori information and the step size is $\alpha = 10^{-5}$. The bottleneck point we try to break through is $I_2^* = 0.2$.

In Fig. 4.3 we compare MMSE and AMFER EXIT charts, when $\mathrm{SNR} = 5.9$ dB. The red curve shows T_1 with AMFER parameters, while the orange curve shows T_1 with MMSE parameters. The blue curve shows T_2^{-1} for the LDPC decoder, which is independent of the parameters. From the figure we can see that in the case of MMSE parameters, the two curves intersect. On the other hand, in the case of AMFER parameters, there is a distinct gap near $I_2^* = 0.2$ between the T_1 curve and the T_2^{-1} curve. We also observe that although the AMFER curve is not superior to the MMSE curve when $I_2 > 0.6$, the ultimate FER is dramatically better; the FER with AMFER is 0.068, as opposed to the FER of 0.605 with MMSE.

In Fig. 4.4, we plot the FER versus SNR for both the MMSE and AMFER parameters. The error bars indicate the 95% confidence interval. At around FER of 10^{-5}, the AMFER parameters outperform MMSE parameters by 0.4 dB.

4.4.2 Single-Track Detection on Ehime waveforms

We test our algorithm on Ehime waveforms. Despite the fact that the waveforms were not created using an error-control encoder, we can still test the turbo detector of Fig. 4.1

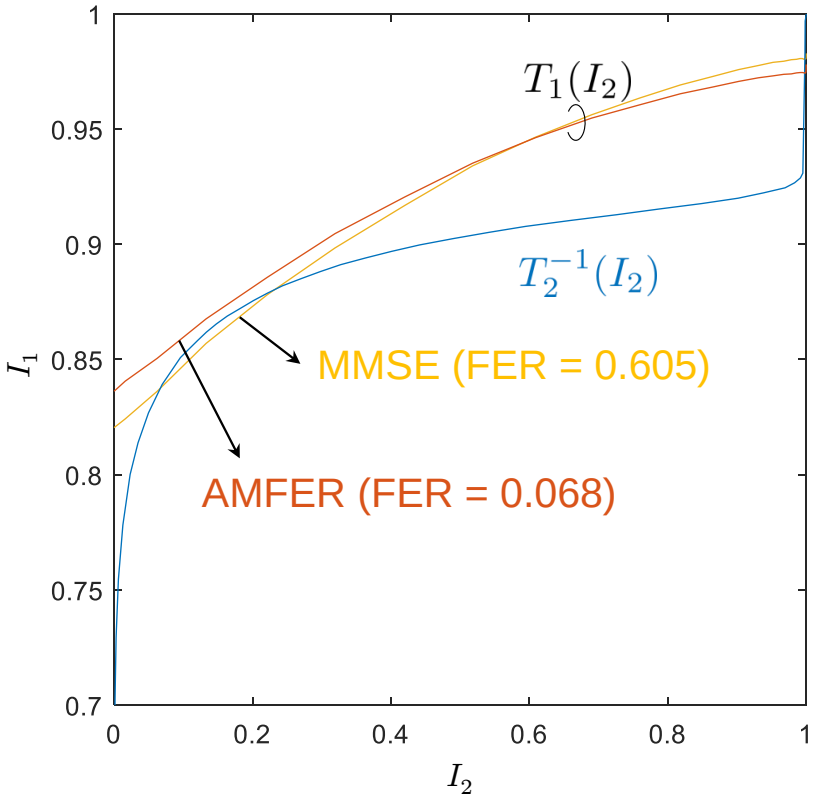


Figure 4.3: EXIT chart with MMSE and AMFER parameters.

through the use of coset leaders; an uncoded block $\mathbf{c}$ of written bits can be interpreted as a codeword in the coset code $\mathbf{c} \oplus \mathcal{C}$ for any linear code $\mathcal{C}$. We also consider the written bits are interleaved and thus an interleaver and a de-interleaver are applied in the system.

We used the first 16200 bits for training the AMFER parameters, and the following 24999 bits to measure FER. Because the codeword length 16200 is much shorter than 24999, we are able to test multiple codewords from a single waveform by looking at different segments of the waveform. In particular, we consider 8800 consecutive bits as different starting locations for 8800 different codewords, and these frames are used for estimating FER. To limit any correlation between consecutive frames, we apply independent white Gaussian electronic noise with zero mean and standard deviation $\sigma_e = 0.04$ to each frame. The corresponding power of the added noise within the Nyquist

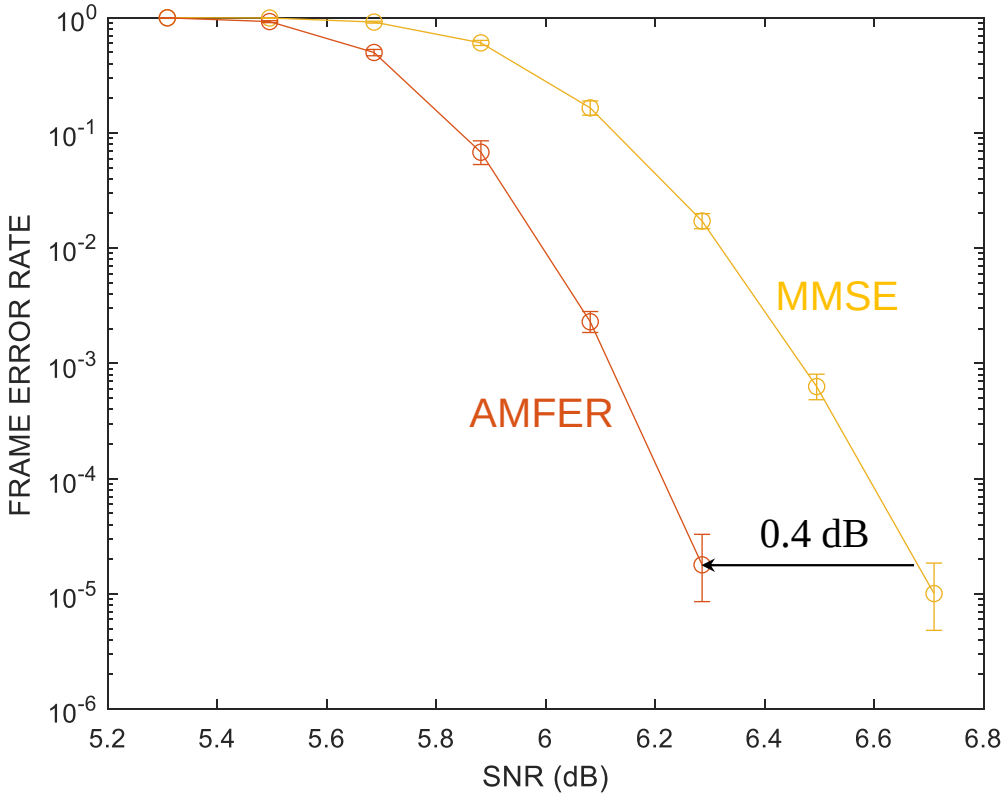


Figure 4.4: FER vs SNR with MMSE and AMFER parameters.

band is 24.6 dB below the saturation (constant response) signal level for the centered 100% width reader at 22.1 nm track pitch, which for that scenario is 10.1% of the total noise power (including media noise) [30].

We test the AMFER algorithm in two scenarios. For the first scenario, the detector is SOVA without any pattern-dependent noise predictor. We consider the case of ten equalizer coefficients and four signal levels. For the second scenario, the detector is SOVA with PDNP. We set ten equalizer coefficients and eight patterns. Each pattern has three parameters (one signal level, one residual noise variance and one noise predictor coefficient). The update equations can be obtained by plugging the PDNP path metric[21, 22] into (4.8) and applying the AMFER algorithm. The AMFER parameters are initialized with MMSE parameters and $I_2^* = 0.3$.

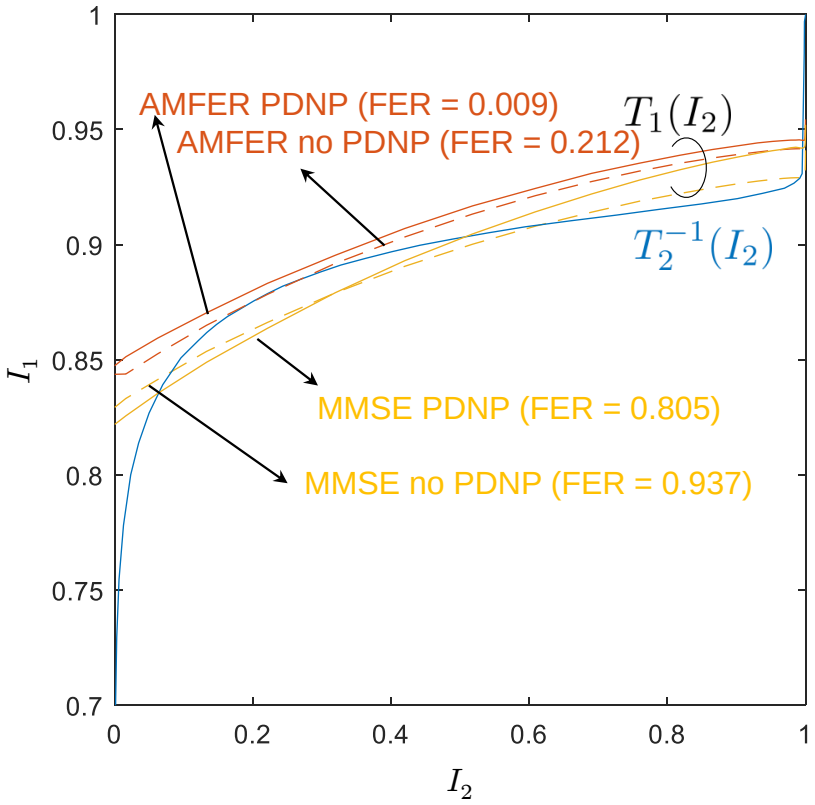


Figure 4.5: EXIT charts based on different detectors.

We compare the EXIT charts for different schemes in Fig. 4.5, assuming the track pitch is 24.1 nm and a centered reader with 70% width. The dashed curves are for the detectors without PDNP, while the solid curves are for the PDNP detectors. The MMSE curves are orange, while the AMFER curves are red. Without PDNP, the MMSE detector achieves FER = 0.937, while the AMFER detector achieves FER = 0.212, an improvement by more than a factor of 4. With PDNP, the MMSE detector achieves FER = 0.805, while the AMFER detector achieves FER = 0.009, an improvement by more than a factor of 89.

The areal density advantage of the AMFER algorithm is illustrated in Fig. 4.6, where we plot FER versus track pitch for the case of a centered reader with 70% width. The legend in this figure is as same as that in Fig. 4.5. The horizontal distance between two curves translates to areal density gain at a given value of FER. From the figure we see that

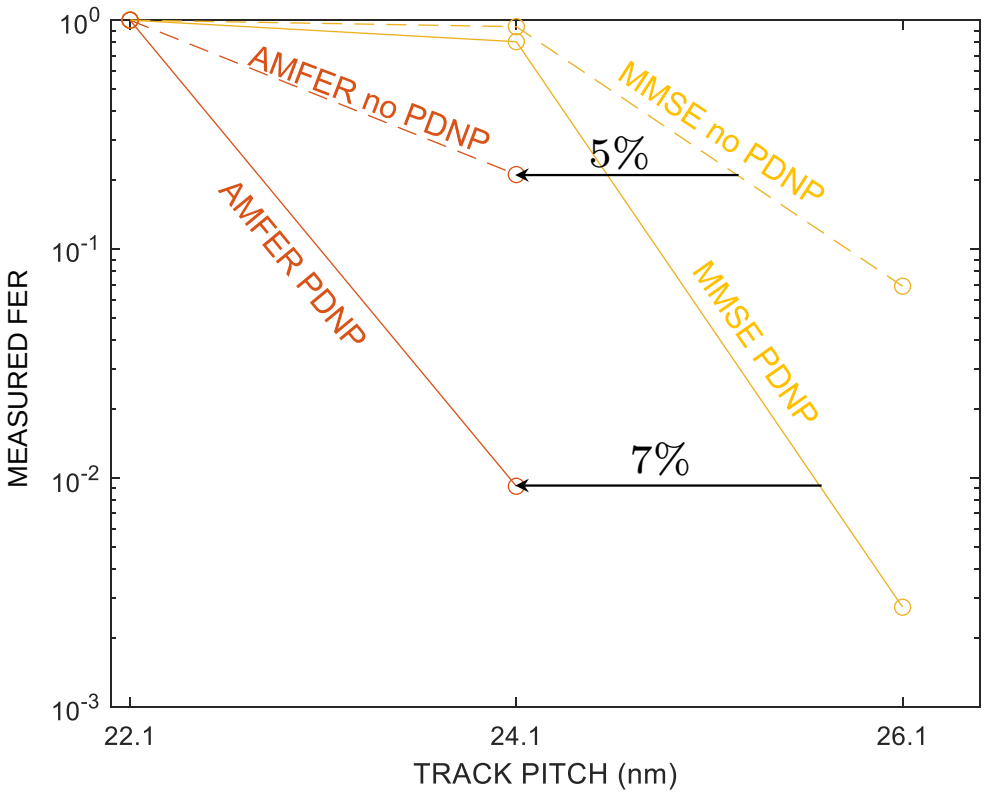


Figure 4.6: Measured FER performance of MMSE and AMFER parameters w/o PDNP.

the areal density advantage of AMFER over MMSE is roughly 5% without PDNP, and it is roughly 7% with PDNP.

4.4.3 Multitrack Detection on Ehime waveforms

We test the AMFER algorithm on a 2D-PDNP SOVA detector with Ehime waveforms through the use of coset leaders. We detect track two and three with eleven matrix-valued equalizer coefficients and sixteen patterns. Each pattern has predictor coefficients $\mathbf{P}_0$ and $\mathbf{P}_1$. The total number of parameters is 188. The LDPC code has a length of 16200 and a code rate of 0.89 from DVB-S2 database. We set the maximum number of decoder iterations to ten and the maximum number of turbo iterations to twenty.

We train the parameters with the first 10000 bits from track two and three. We consider

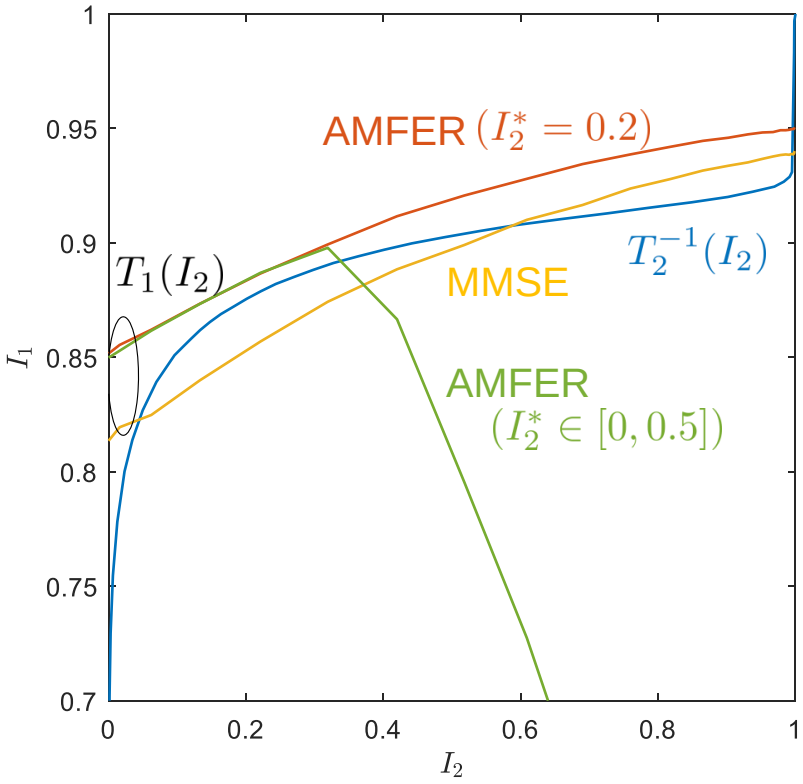
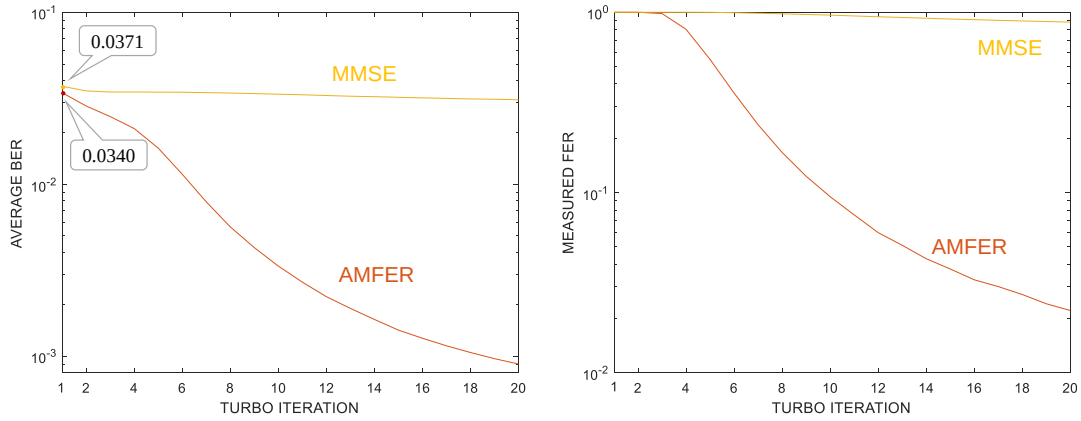


Figure 4.7: EXIT chats with MMSE parameters (yellow), AMFER parameters trained by $I_2^* = 0.2$ (green) and AMFER parameters trained by $I_2^* \in [0, 0.5]$.

23100 consecutive bits as different starting locations for 23100 different codewords, and use these frames for estimating FER. We apply independent electronic noise ($\mathrm{SNR}_e = 24.6$ dB) to each frame to reduce correlation between consecutive frames. First we test on the track pitch of 24.1 nm with a 70% reader and an 85% reader, centered on track two and three. We plot the yellow T_1 curve with MMSE parameters and blue T_2^{-1} curve based on the LDPC code in Fig. 4.7. The two curves intersect, which leads to a high measured FER of 0.88. We then train at the $I_2^* = 0.2$ where we try to break through and plot the T_1 curves with the trained AMFER parameters plotted in green. As the figure shows the curve opens a tunnel around 0.2 but drops rapidly after 0.3. The result is different from what we get in single-track detection because the parameter space is small in single-track case so that small changes at one point will not cause big changes at other places. However, we do not



(a) (b)

Figure 4.8: (a) Average BER measured by the sign of $\boldsymbol{\lambda}_1$ as turbo iteration increases. (b) Evolution of measured FER as turbo iteration increases.

want to put our efforts in all places of I_2 because the bottleneck area is the key to lowering the FER. One way to fix this problem is to train with I_2^* evenly sampled from 0 to 0.5. We plot the T_1 curve with the new AMFER parameters in red. As we can see a clear tunnel from left to right, and the measured FER = 0.022, an improvement by a factor of 40 over MMSE parameters. However, in practice we do not have to constrain the detector to use the same set of parameters during the whole turbo detection.

Next we investigate the relationship between BER and FER. We plot average BER measured by the sign of $\boldsymbol{\lambda}_1$ versus the turbo iteration in Fig. 4.8 (a). The yellow MMSE curve drops very slowly as opposed to the red AMFER curve. We also label the BER of the first iteration for each curve. Since there is no a priori information, the BER of the detection actually only depends on the equalized signal. The lower BER of AMFER indicates AMFER parameters can help reduce the BER for hard-output detection. In Fig. 4.8 (b) we plot the measured FER versus turbo iteration. We find that the FER curves have the same trend as BER curves. When $z_{1,k} = a_k \lambda_{1,k} \sim \mathcal{N}(\mu_1, 2\mu_1)$, we can get BER

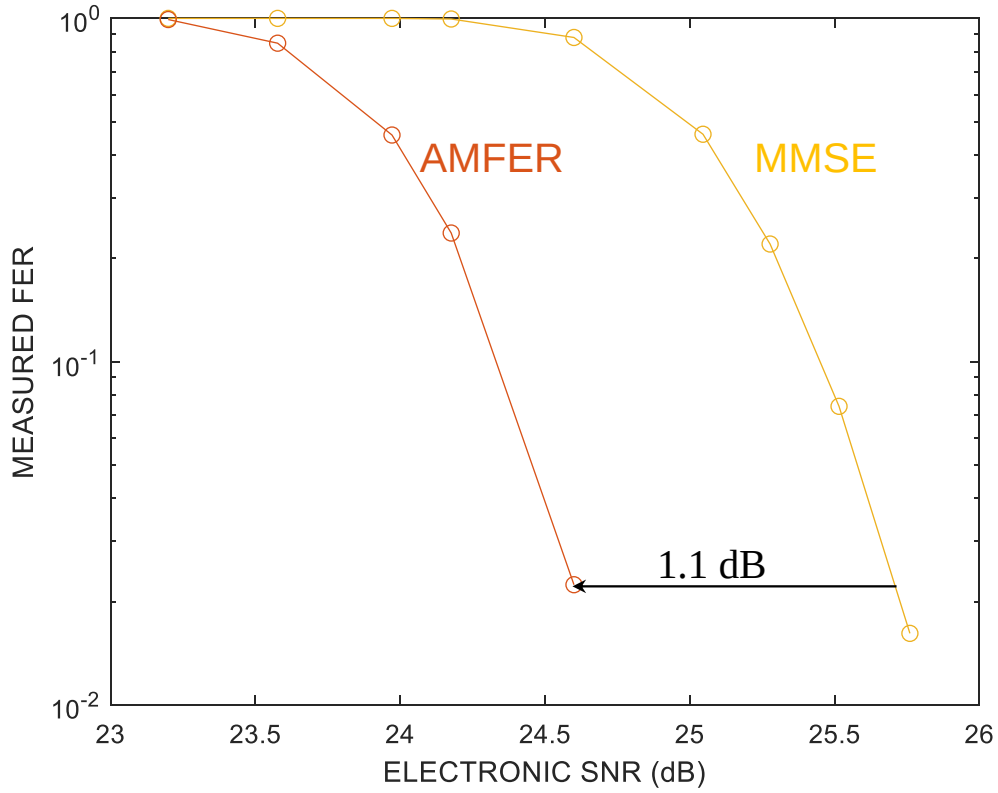


Figure 4.9: Measured FER vs electronic SNR with MMSE and AMFER parameters.

= $P(z_{1,k} < 0) = Q(\sqrt{\mu_1/2})$. This equation means when BER decreases, μ_1 will increase, which leads to the increase of I_1. Eventually, the FER will decrease. However, in practice $z_{1,k}$ is not consistent Gaussian and thus a lower BER cannot guarantee a lower FER. One counter example is AMBER parameters.

We also test the measured FER under different electronic noise shown in Fig. 4.9. The red curve represents AMFER parameters while the yellow curve represents MMSE parameters. Given the FER of 0.02, we observe that the AMFER parameters can resist 1.1 dB more electronic noise than the MMSE parameters.

Lastly, we test the measured FER at different track pitches in Fig. 4.10 with two 70% centered readers. At 26.1 nm, all 23100 testing frames are decoded successfully with AMFER parameters. To quantify the areal density gain, we plot the boundary area (gray triangle) where the line from 24.1 nm to 26.1 nm could appear. At the BER of 10^{-2},

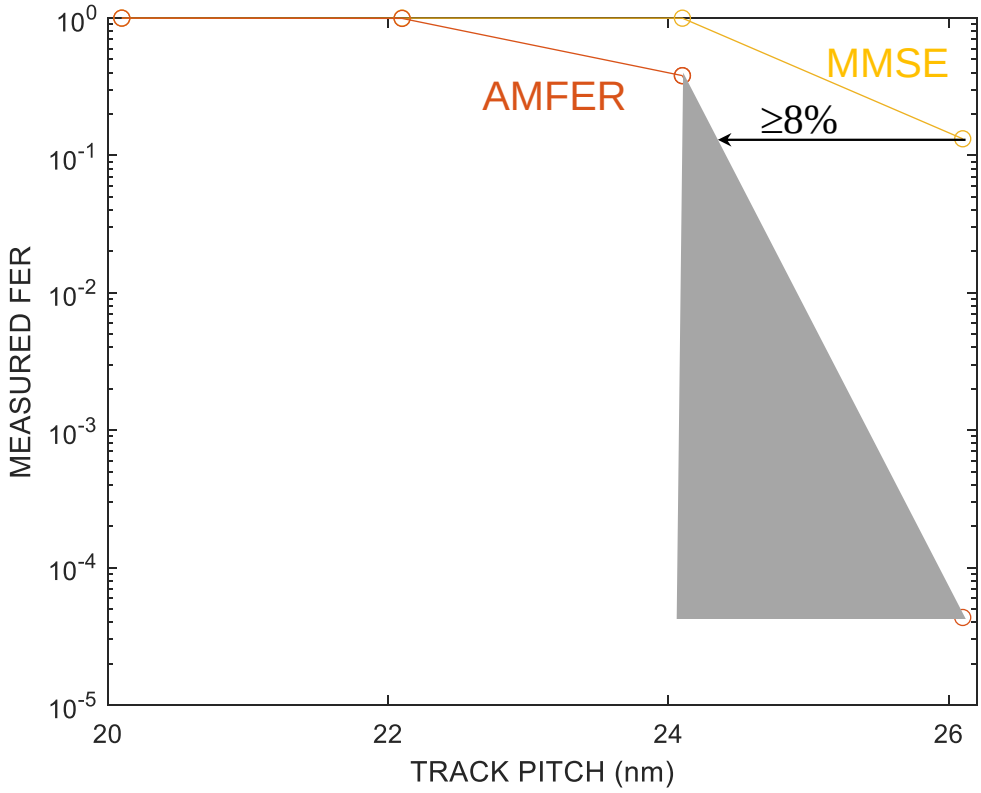


Figure 4.10: Measured FER vs track pitch with MMSE and AMFER parameters.

AMFER parameters achieve at least 8% areal density gain over MMSE parameters. The same method shown in Section 3.4.3 can be applied to avoid redundant or negligible parameters.

4.5 Summary

In this chapter we demonstrate the close connection between the FER and EXIT charts, and propose a strategy for adapting the parameters of an equalizer and a soft-output detector so as to minimize the FER. The AMFER algorithm is general and applies to a wide range of detectors and read channels, including those using multiple readers and multitrack detection. For the cases of single-track detection and multitrack detection, numerical results using realistic readback waveforms show that the parameters found by the AMFER algorithm can lead to dramatic improvements in FER and areal density

compared to MMSE parameters.

CHAPTER 5

NEURAL NETWORK FOR MAGNETIC RECORDING

5.1 Introduction

In previous chapters, we stick to the traditional signal processing techniques. In this chapter, we will apply neural network techniques to deal with the magnetic readback waveforms. The object of this chapter is to construct and design a neural network to mimic a hard-output detector, and compare the performance and complexity to traditional methods.

This chapter is organized as follows. In Section 5.2, we review some mature neural network structures. In Section 5.3, we design a neural network for hard-output detection. We evaluate the performance on Ehime waveforms, and compare it to the AMBER algorithm. In Section 5.4, we summarize this chapter.

5.2 Classic Neural Network Structures

The artificial neural network is a subset of machine learning, loosely inspired by the biological brain, first proposed in 1943 by Warren McCulloch and Walter Pitts [61].

A typical neural network consists of an input layer, one or more hidden layers and one output layer, as shown in Fig. 5.1. Deep learning is a neural network with three or more layers (exclude the input layer). Each layer has several neurons. Each neuron expect in the input layer is a microprocessor, which collects all the data from all neurons in the previous layer, does calculations, and transfers one result to the next layer (except the output layer). Denote vector $\mathbf{a}_0$ (also written as $\mathbf{x}$) to be the data of the input layer. Assume the ith layer

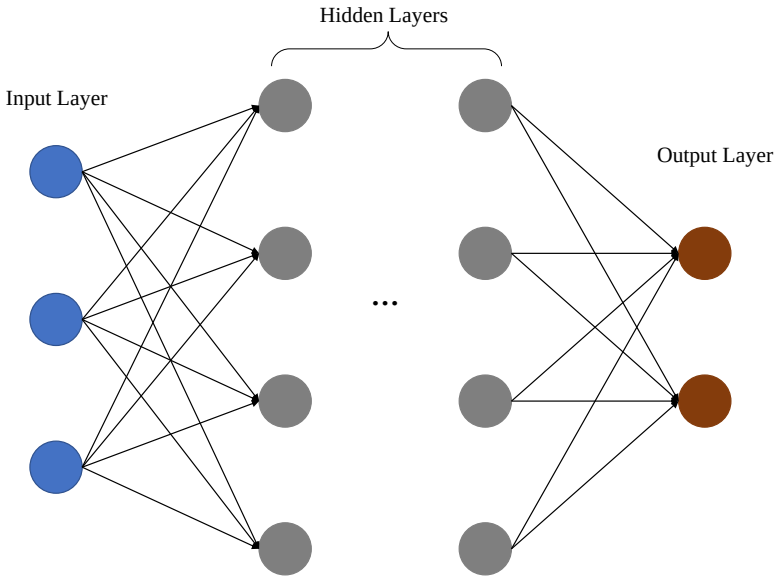


Figure 5.1: A typical structure of the artificial neural network: one input layer (labeled in blue), one or more hidden layers (labeled in grey) and one output layer (labeled in red). Circles represent neurons and arrows represent the direction of data flow.

has N_i neurons. The output data of ith hidden layer is calculated by

$$\mathbf{a}_i = g(\mathbf{W}_i^T \mathbf{a}_{i-1} + \mathbf{b}_i), \tag{5.1}$$

where $\mathbf{W}_i = [\mathbf{w}_{i,0},\ ...,\ \mathbf{w}_{i,N_{i-1}}]$ and $\mathbf{b}_i = [b_{i,0},\ ...,\ b_{i,N_{i-1}}]^T$. $\mathbf{w}_{i,j}$ and $b_{i,0}$ are associated with the calculation for the jth neuron in ith layer. $g(\cdot)$ represents a nonlinear activation function. Commonly used activation functions are listed as follows. ReLU:

$$g(z) = \max(0, z). \tag{5.2}$$

Sigmoid function (also written as $\sigma(\cdot)$):

$$g(z) = \frac{1}{1 + e^{-z}}. \tag{5.3}$$

Hyperbolic tangent function:

$$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}. \tag{5.4}$$

The output layer has the same structure as the hidden layer but the activation function is chosen according to users' demand. The linear, sigmoid and softmax activation functions are the typical choices. The linear activation means no changes on the result, which fits the case that the output value is in the interval $(-\infty, +\infty)$. The sigmoid function is usually used to measure probability since it ranges from 0 to 1. The softmax function is given by

$$\mathrm{softmax}(z_i) = \frac{e^{z_i}}{\sum_j e^{z_j}}. \tag{5.5}$$

Since the sum softmax outputs is one, it is mainly used in classification problems where $\mathrm{softmax}(z_i)$ measures the probability of the ith class.

The difference between neural network output $\hat{\mathbf{y}}$ and real output $\mathbf{y}$ is measured by a loss function $L(\hat{\mathbf{y}}, \mathbf{y})$. The cost function is defined as the average loss of all training sets:

$$J(\mathbf{W}_1, \mathbf{W}_2, ..., \mathbf{b}_1, \mathbf{b}_2, ...) = \frac{0}{N_t - 1} \sum_{i=1}^{N_t} L(\hat{\mathbf{y}}_i, \mathbf{y}_i), \tag{5.6}$$

where N_t is the number of training sets and the cost function needs to be differentiable. In the training process, the backpropagation algorithm [62] computes the gradient of the cost function with respect to each weight efficiently. The efficiency makes it easy to apply gradient methods such as gradient descent and Adam [63].

According to the universal approximation theorem [64, 65], a neural network with linear output and sigmoidal activation function given enough hidden neurons can approximate any continuous function on a closed and bounded subset of $\mathbb{R}^n$[66]. But the required number of neurons can be infeasible large. In some cases deeper neural networks can reduce the number.

Deep learning is not newly invented but becomes very popular recently because large

and deep neural networks driven by big data can achieve better performance than small neural networks, as well as the development of powerful computers and machine learning algorithms.

A convolutional neural network [67, 68, 69] is a special neural network for processing data with a grid-like structure, such as time-series data and image data. Commonly used layers of CNN's are convolutional layers, pooling layers and fully connected layers. A CNN must have one or more convolutional layers that perform a bunch of convolution operations. The pooling layers perform maximizing or averaging operations on the data. The fully connected layers are the usual neural network layers shown in Fig. 5.1. Convolutional layers can save tons of parameters than fully connected layers because one convolution block can work for lots of data segments and each output value may depend only on a small number of input data.

Besides, a recurrent neural network [70] is another neural network for processing sequential data, which has achieved great success in natural language processing and speech processing. The structure of a standard RNN is shown in Fig. 5.2. At ith stage, the information transferred to the next stage and the output $\hat{y}_i$ are calculated as

$$\begin{aligned} \mathbf{a}_i &= g_1(\mathbf{w}_a^T \begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{x}_i \end{bmatrix} + \mathbf{b}_a) \\ \hat{y}_i &= g_2(\mathbf{w}_y^T \mathbf{a}_i + \mathbf{b}_y), \end{aligned} \tag{5.7}$$

where $g_1(\cdot)$ and $g_2(\cdot)$ are activation functions. Each stage shares the same parameter set $[\mathbf{w}_a, \mathbf{b}_a, \mathbf{w}_y, \mathbf{b}_y]$. In some sophisticated cases, there could be more layers between the input and output layers.

Gradients propagated over many stages in RNN's could vanish so that the model cannot deal with the data with long-term interactions. The long short-term memory (LSTM) layer [71] improves the standard RNN with feedback connections to avoid vanished gradient problems [66]. The structure of the LSTM cell is shown in Fig. 5.3 [72]. A typical LSTM

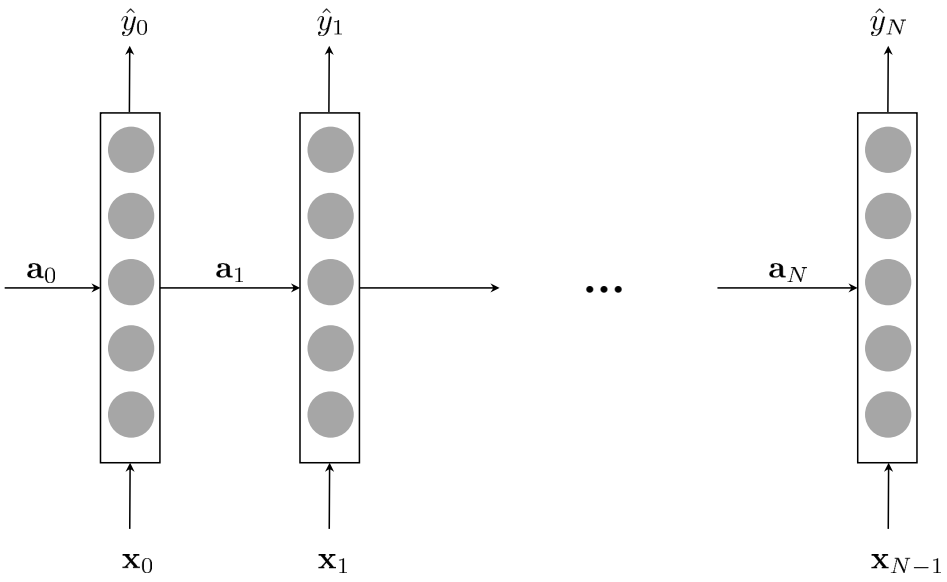


Figure 5.2: The standard recurrent neural network.

cell has three types of gates: forget gate, denoted as $\mathbf{\Gamma}_f$; update gate, denoted as $\mathbf{\Gamma}_u$; and output gate, denoted as $\mathbf{\Gamma}_o$. The cell state at ith stage, denoted as $\mathbf{c}_i$, is like a conveyor belt running down the entire chain. Adding or removing information on the belt is controlled by the forget gate and update gate. The forget gate $\mathbf{\Gamma}_f$ is calculated by

$$\mathbf{\Gamma}_f = \sigma(\mathbf{W}_f^T \begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{x}_i \end{bmatrix} + \mathbf{b}_f), \tag{5.8}$$

where $\mathbf{a}_{i-1}$ is the previous hidden state and $\mathbf{x}_i$ is the current input. The sigmoid activation makes its output between zero and one. One means completely remember and zero means completely forget. In practice, most results will be close to the two extremes. Then the update gate will decide whether and what new information will be added to the belt, which

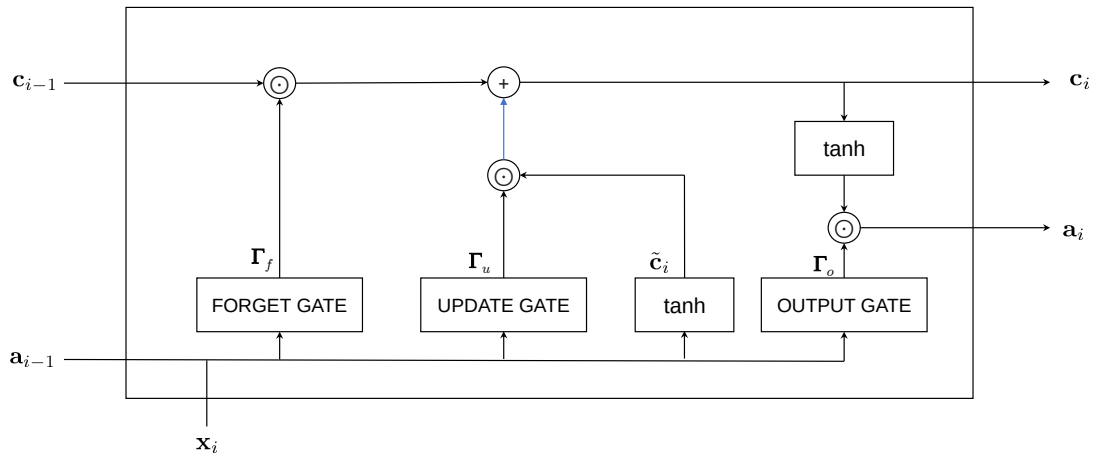


Figure 5.3: Data flow inside a long short-term memory cell.

is calculated by

$$\begin{aligned} \mathbf{\Gamma}_u &= \sigma(\mathbf{W}_u^T \begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{x}_i \end{bmatrix} + \mathbf{b}_u), \\ \hat{\mathbf{c}}_i &= \tanh(\mathbf{W}_c^T \begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{x}_i \end{bmatrix} + \mathbf{b}_c). \end{aligned} \tag{5.9}$$

The final update equation of the new cell state is

$$\mathbf{c}_i = \mathbf{\Gamma}_f \odot \mathbf{c}_{i-1} + \mathbf{\Gamma}_u \odot \hat{\mathbf{c}}_i, \tag{5.10}$$

where $\odot$ is Hadamard product (elementwise product). The update for the new hidden state

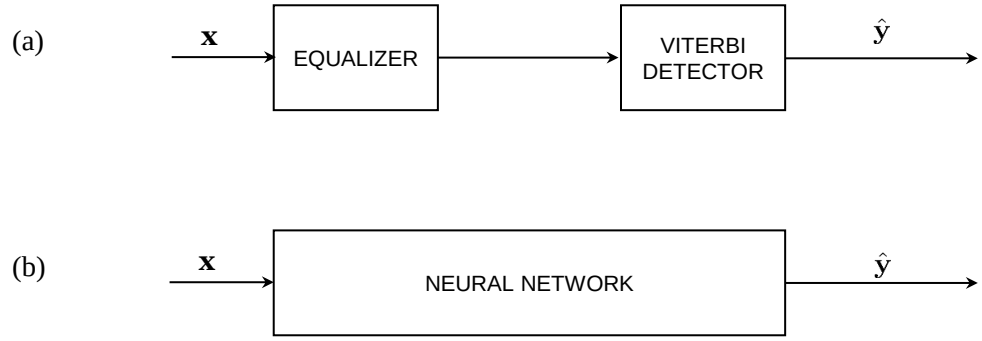


Figure 5.4: (a) A conventional detection system. (b) A neural network detection system.

$\mathbf{a}_i$, which is also used as the stage output, is calculated by

$$\begin{aligned} \mathbf{\Gamma}_o &= \sigma(\mathbf{W}_o^T \begin{bmatrix} \mathbf{a}_{i-1} \\ \mathbf{x}_i \end{bmatrix} + \mathbf{b}_o) \\ \mathbf{a}_i &= \mathbf{\Gamma}_o \odot \tanh(\mathbf{c}_i). \end{aligned} \tag{5.11}$$

5.3 Neural Network for Hard-Output Detection

The conventional magnetic recording system consists of an equalizer and a sequence detector as shown in Fig. 5.4 (a). The readback waveform vector $\mathbf{x}$ is equalized corresponding to a target. The equalized signal and the target are involved in the maximum likelihood detection, performed by the Viterbi algorithm. A neural network detection does not need a cascade structure. Instead, the system takes into the readback waveform and outputs the estimated bits $\hat{\mathbf{y}}$ directly, as shown in Fig. 5.4 (b).

We design a recurrent neural network to mimic sequence detection. The layers of the RNN is shown in Fig. 5.5. Even though there is a convolutional layer in the network, we still call it a recurrent neural network since the recurrent layer is the core layer. The input

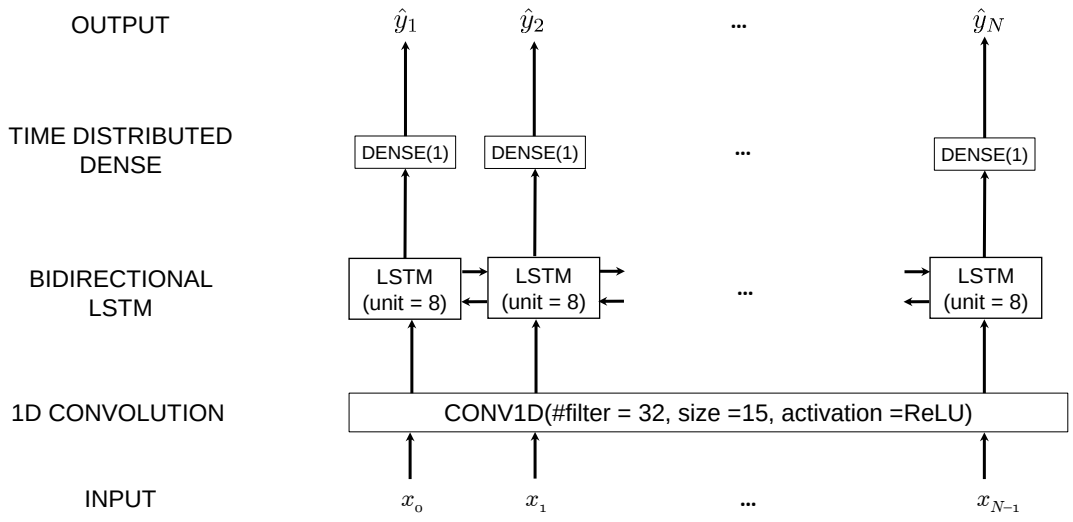


Figure 5.5: The recurrent neural network aims to do sequence detection. The network consists of a convolution layer, a bidirectional LSTM layer and a time-distributed dense layer.

layer is formed by an $N \times 1$ vector of sampled waveforms. The first hidden layer is one-dimensional convolutional layer. This layer aims to extract common features shared during the sequential input. Note that the convolution operation in machine learning does not reverse the operand, which is different in signal processing. In this chapter, we will stick to the version of machine learning. In this layer, there are 32 length-15 sequences convolving with the input vector and the activation function is ReLU shown in (5.2). The output is an $N \times 32$ matrix as

$$\mathbf{a}_1 = g\left(\begin{bmatrix} \mathbf{x} * \mathbf{c}_0 \\ \dots \\ \mathbf{x} * \mathbf{c}_{31} \end{bmatrix}^T\right). \tag{5.12}$$

The second hidden layer is a bidirectional LSTM (BLSTM) layer, which has another LSTM chain running backward. We denote the ith forward output and backward output as $\mathbf{a}_i^f$ and

Table 5.1: Output size and number of parameters of each layer in the RNN shown in Fig. 5.5.

Layer	Output Size	Param #
Input	(N,1)	0
1D Convolution	(N,32)	512
BLSTM	(N,16)	2624
Time-Distributed Dense	(N,1)	17
Output	(N,1)	0

$\mathbf{a}_i^b$. The layer aims to attack the media noise since the current bit suffers from the noise from both forward and backward directions. The third hidden layer is a time-distributed dense layer. This layer applies the same densely connected net to every time step during LSTM cells unrolling. Sigmoid activation function is applied to measure the probability that $\hat{y}_i = 1$, denoted as p_i. The equation is written as

$$p_i = \sigma(\mathbf{w}_a^T \begin{bmatrix} \mathbf{a}_i^f \\ \mathbf{a}_i^b \end{bmatrix} + b_a). \tag{5.13}$$

The hard decision that $\hat{y}_i$ = 0 or 1 is made according to whether p_i is smaller or larger than 0.5. The output size and number of parameters required are listed in Table. 5.1. The total required number of parameters is 3153.

For the training and testing we still use the Ehime waveforms as introduced in Section 2.5.1. We add electronic noise with $\sigma = 0.04$ on the readback waveforms of track three. We use the first 16000 bits for training and validation and the rest bits for testing. Since the training data is limited, we truncate 1000 bits and shift sixteen bits to form a new training frame. In total we create 1000 frames and add independent electronic noise to reduce the dependence of neighbouring frames.

There is no differentiable cost function for bit-error rate as explained in Chapter 3. One alternative cost function is the cross entropy loss if we view the detection as a binary

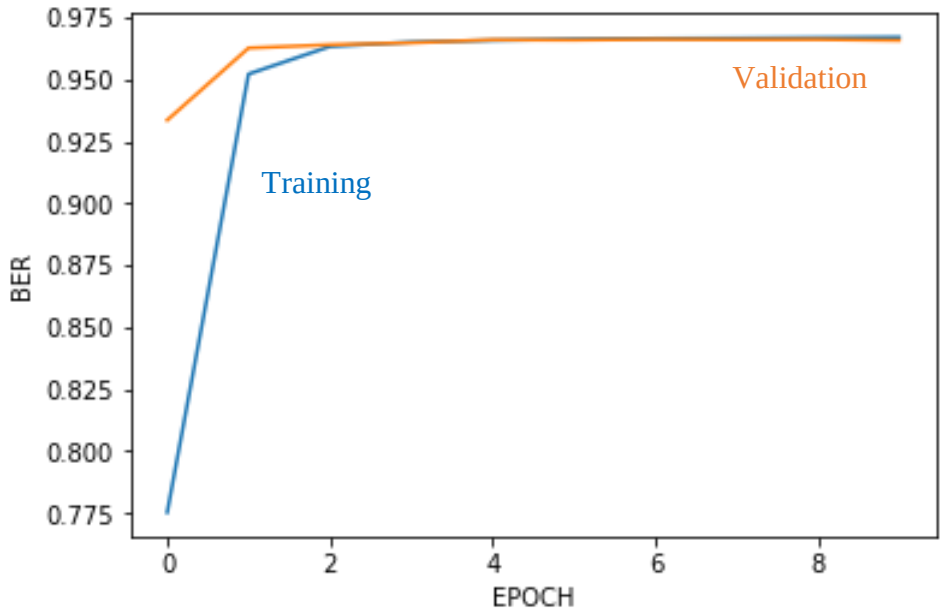


Figure 5.6: The BER evolution of training and validation data as the epoch increases.

classification problem. The cost function is defined as

$$J(\mathbf{\Theta}) = -\frac{1}{N_t}\sum_{i=0}^{N_t-1} y_i \log p_i + (1-y_i)\log(1-p_i), \tag{5.14}$$

where $\mathbf{\Theta}$ represents all the weights and bias to be optimized in (5.12)-(5.13) and the bidirectional LSTM layer. When the predicted probability approaches to the wrong bit, the log loss increases sharply. The log loss slowly decreases as the predicted probability approaches the correct bit. In other words, the log loss especially penalizes those predictions that are confident but actually wrong. However, we notice that lower log loss not necessarily means a lower BER. For example, suppose three bits $[-1,-1,-1]$ has two sets of probabilities: $[0.1, 0.6, 0.6]$ and $[0.4, 0.4, 0.4]$. The first set has one bit error with a log loss of 1.11, while the second set has three bit errors with a log loss of 0.92.

We apply the Adam optimization algorithm to minimize the cost function and use a mini-batch of size sixteen to accelerate the convergence. 900 frames are used to train the model parameters and the rest 100 frames are for validation. The BER of training and

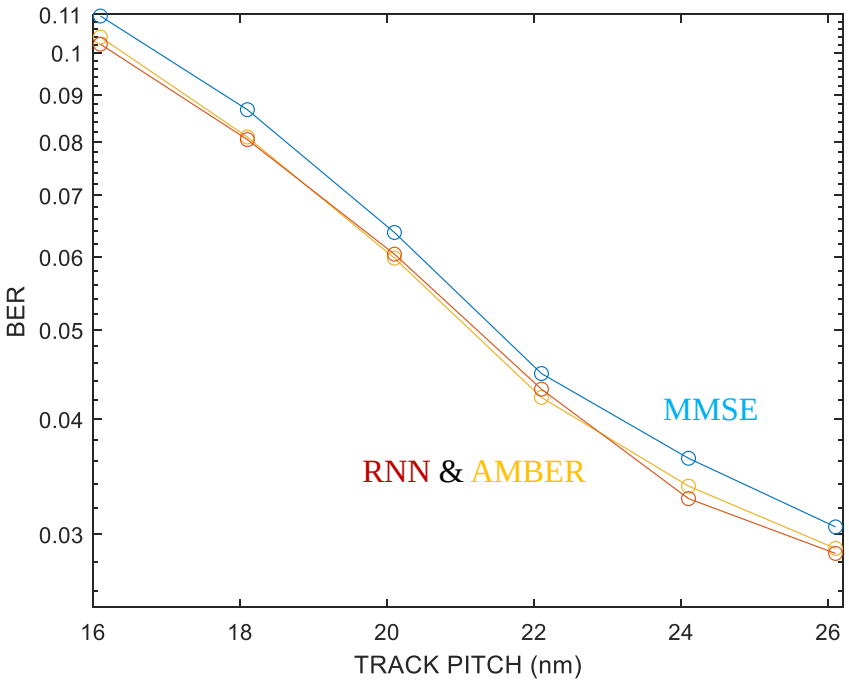


Figure 5.7: BER performance of the RNN and conventional structures (with MMSE and AMBER parameters).

validation data as the epoch increases is shown in Fig. 5.6. The blue curve represents the training data, while the orange curve represents the validation data. Two curves converge after the fourth epoch and the narrow gap between the two curves indicates that there is little overfitting in the model. Next we transfer the weights of trained model to a long model for testing. We compare the performance of the RNN and conventional structure in Fig. 5.7. We plot the RNN performance in red and the conventional structure with MMSE PDNP and AMBER PDNP parameters in blue and orange. The MMSE PDNP is based on twenty equalizer coefficients and eight patterns with one predictor coefficient and AMBER PDNP is based on twenty equalizer coefficients and four patterns with one predictor coefficient. Results show that the RNN outperforms MMSE by approximately 4% area density gain. The RNN achieves a similar performance as AMBER parameters, but AMBER only needs to train 32 parameters. We increase the depth of the RNN with more than one BLSTM

layer but cannot achieve further gain.

5.4 Summary

In this chapter, we design a neural network for single-track hard-output detection. The network achieves similar BER performance with the traditional detector with AMBER parameters and is better than MMSE parameters. However, the network requires much more parameters than the traditional detectors. The extension to multitrack detection is straightforward.

CHAPTER 6
CONCLUSIONS AND FUTURE WORK

This dissertation proposes signal processing algorithms for magnetic recording. In particular, a multitrack detection structure and the multitrack detector with 2D-PDNP for TDMR is proposed. This thesis also proposes three strategies to choose parameters. The first strategy is a conventional closed-form solution based on the MMSE criterion. The second strategy is an adaptive algorithm to choose parameters aiming to minimize the BER after detection. The third strategy is to tune parameters to minimize the FER when the detector works with an error-control decoder in a turbo fashion. The thesis also designs a neural network to do the detection. The thesis reports the results of the proposed algorithms on simulated waveforms. In this chapter, we summarize all the contributions of this thesis and then propose a few directions where this work can be extended and improved.

6.1 Contributions

1. In Chapter 2, we proposed the multitrack pattern-dependent and autoregressive model for 2D media noise and the joint maximum-likelihood solution to the detection. The multitrack PDNP can predict current noise via the noise from both crosstrack and downtrack, so as to mitigate 2D media noise. Bulks of parameters are required in the multitrack detection system. The equalizer coefficients are chosen to minimize the MSE between equalized signals and target output. We presented two strategies for choosing multitrack PDNP parameters. One requires training: the parameters for each pattern are chosen to minimize the sum mean-squared prediction error in the training process. The other does not require training: The parameters are estimated by reliable bits from a PDNP BCJR detector.

The detection and estimation work iteratively until convergence. Numerical results on simulated waveforms show PDNP parameters from both strategies outperform non-PDNP Viterbi detectors.

2. In Chapter 3, we considered the problem of tuning the PDNP parameters to minimize the BER instead of MSE. In Section 3.2, we proposed a cost function based on the path metric margin. We discovered that minimizing the cost function is equivalent to minimizing the probability of detection errors when the tail of the margin PDF is exponential. We developed the AMBER algorithm by applying a stochastic gradient on the cost function. To further improve the algorithm, we proposed strategies to choose the threshold and the step size in the algorithm in Section 3.3. We implemented the algorithm on both single-track and multitrack detection, and numerical results show AMBER parameters can reduce the BER effectively compared with MMSE parameters. In Section 3.4.3, we proposed to optimize the parameter space by adapting part of the parameters via the AMBER algorithm.

3. In Chapter 4, we brought the error-control coding into the picture. We tuned the PDNP parameters to improve the quality of extrinsic LLR's so as to minimize the FER after decoding. We used mutual information to measure the quality of soft information and exploited the close connection between the FER and the gap between the two curves in an EXIT chart. We proposed a cost function and the AMFER algorithm to lift the curve produced by the detector given a fixed error-control code. We verified the AMFER algorithm on a simple ISI channel, as well as single-track and multitrack detection systems, and numerical results revealed that AMFER parameters can reduce the FER significantly over MMSE parameters.

4. In Chapter 5, we designed a neural network to replace the detector. Numerical results demonstrated that the network has a better performance than MMSE parameters and

a similar performance from AMBER parameters. We also revealed that the neural network requires much more parameters compared to traditional systems.

6.2 Future Work

All the experiments in Chapter 4 are based on an off-the-shelf length-16200 rate-8/9 irregular LDPC code that was designed for television broadcast (DVB-S2), and is not optimized for magnetic recording. Interestingly, the area below the two curves in an EXIT chart can be used to estimate the code rate R and channel capacity C, respectively [73, 74]:

$$R \approx \int_0^1 T_2^{-1}(x)dx,$$
$$C \approx \int_0^1 T_1(x)dx.$$

Shannon's theorem enables reliable communication when $R < C$. The above integrals imply that, in terms of the EXIT chart, we need the area under the decoder curve to be less than the area under the detector curve. Any space between the curves is thus an opportunity to improve performance: If we are able to exploit the gap to somehow move the decoder curve closer to the detector curve, without intersecting, we will have effectively designed a better LDPC code with a higher code rate that is matched to the specific characteristics of the soft-output detector. A promising approach to achieving the above-described goal of optimizing the LDPC code to match the channel transfer function is the curve-fitting approach of [75, 49].

In the bigger picture of overall read-channel design, AMFER can be viewed like this: given an LDPC code, AMFER optimizes the detector. In prior work, others have solved the complementary problem: given a channel, optimize the LDPC code. We can investigate the intriguing possibility of jointly optimizing the soft-output detector and the LDPC decoder using a combination of the AMFER concept and curve fitting. A promising candidate for

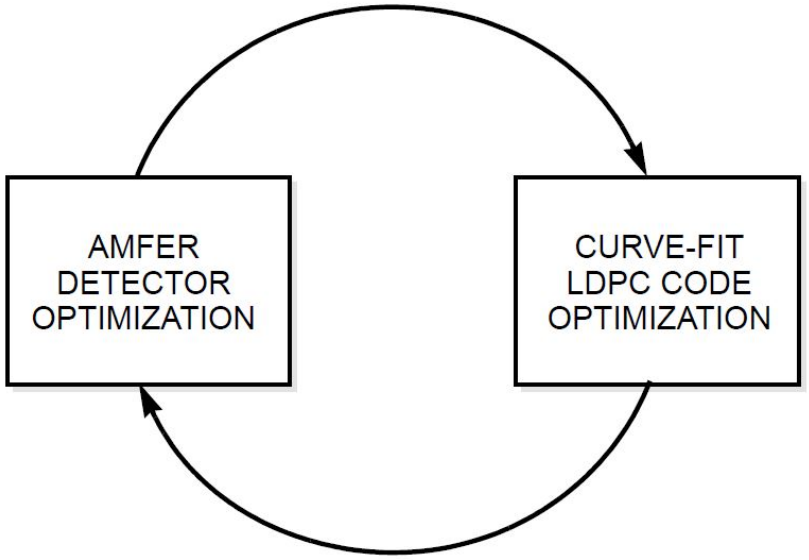


Figure 6.1: An iterative approach to jointly optimizing an LDPC code and a soft-output detector.

joint optimization is the iterative strategy shown in Fig. 6.1, where for each iteration curve fitting is used to optimize the LDPC code for a given soft-output detector, and then the soft-output detector is optimized via AMFER for that given code. In terms of the detector and decoder curves in the EXIT chart framework, one can imagine that these two curves are alternatively being optimized to maintain a gap (ensuring low FER) while maximizing the area under them (ensuring a high capacity and high code rate). The end result could be a high-rate code with low FER, and a potentially big payoff in terms of areal density gains.

Another direction is to bring modulation coding [76] into the framework especially in the 2D case. The modulation coding can avoid bit patterns that cause errors more frequently at the cost of data redundancy. A set of carefully generated modulation codes could bring higher areal density against existing read channels.

In Chapter 5, we designed a neural network for a hard output detector. Neural networks can also function as a soft-output detector and be used in turbo detection. The input of the neural network will be the sampled readback waveforms and the a priori

information from the decoder. The output will be the extrinsic LLR's which are used as the a priori information for the decoder. The cost function of the neural network can be the same cost function we present in Chapter 4. We apply the AMFER algorithm on SOVA detectors instead of BCJR detectors in Chapter 4 because the equation of SOVA is easy to analyze, even though BCJR detectors are maximum a posteriori detectors and can achieve better performance than SOVA. We think neural networks have the potential to outperform AMFER SOVA detectors and provide further areal density gain.

Appendices

APPENDIX A

CONVERGENCE CONDITION AND PROOF OF ALGORITHM 3

Denote $T(\tau) = \frac{1}{b(\boldsymbol{\Theta}(\tau))}$ and then $\tau_{n+1} = T(\tau_n)$. According to Banach fixed point theorem [77], the convergence condition of this algorithm is: there exists $q \in [0, 1)$ such that

$$|T(x) - T(y)| < q|x - y|. \tag{A.1}$$

Proof.

$$\begin{aligned} |T(\tau_n) - T(\tau_{n-1})| &< q|\tau_n - \tau_{n-1}| \\ |\tau_{n+1} - \tau_n| &< q|T(\tau_{n-1}) - T(\tau_{n-2})| \\ &< q^2|\tau_{n-1} - \tau_{n-2}| \\ &\dots \\ &< q^n|\tau_1 - \tau_0|. \end{aligned} \tag{A.2}$$

Let $\epsilon > 0$ be arbitrary, since $q \in [0, 1)$, we can find a large $n \in \mathbb{N}$ so that

$$q^n|\tau_1 - \tau_0| < \epsilon. \qquad \square$$

APPENDIX B

DERIVATION OF (4.1)

The mutual information between a discrete random variable $a \in \mathcal{A}$ and continuous random variable λ is

$$I = \sum_{a \in \mathcal{A}} \int_{-\infty}^{+\infty} P(a, \lambda) \log_2 \frac{P(\lambda|a)}{P(\lambda)} d\lambda.$$

When $P(a = -1) = P(a = 1) = 1/2$ and $P(\lambda|a) = \frac{1}{\sqrt{2\pi 2\mu}} e^{-\frac{(\lambda - a\mu)^2}{2(2\mu)}}$, I reduces to:

$$\begin{aligned} I &= \sum_{a \in \{-1,1\}} \int_{-\infty}^{+\infty} -P(a, \lambda) \log_2 \frac{1 + e^{-a\lambda}}{2} d\lambda \\ &= 1 - E(\log_2(1 + e^{-a\lambda})). \end{aligned}$$

REFERENCES

[1] J. Hale, *More Than 500 Hours Of Content Are Now Being Uploaded To YouTube Every Minute*, https://www.tubefilter.com/2019/05/07/number-hours-video-uploaded-to-youtube-per-minute/, Accessed: 2019-08-23.

[2] D. Reinsel, J. Gantz, and J. Rydning, "The digitization of the world: From edge to core," *Framingham: International Data Corporation*, 2018.

[3] J. Miller, *Cloud Storage Is Much More Secure Than You Think*, https://www.forbes.com/sites/quora/2017/09/18/cloud-storage-is-much-more-secure-than-you-think/#3e253a867403, Accessed: 2019-09-02.

[4] S. Dahandeh, F. Erden M, and R. Wood, "Areal-density gains and technology roadmap for two dimensional magnetic recording," in *The 26th Magnetic Recording Conference*, Aug. 2015, pp. 85–86.

[5] E. Soljanin and C. N. Georghiades, "On coding in multi-track, multi-head, disk recording systems," in *Proceedings of GLOBECOM'93. IEEE Global Telecommunications Conference*, IEEE, 1993, pp. 18–22.

[6] R. Wood, R. Galbraith, and J. Coker, "2-D magnetic recording: Progress and evolution," *IEEE Transactions on Magnetics*, vol. 51, no. 4, pp. 1–7, 2015.

[7] A. Shilov, *Western Digital Launches Ultrastar DC HC530 14 TB PMR with TDMR HDD*, https://www.anandtech.com/show/12665/western-digital-launches-ultrastar-dc-hc530-14-tb-pmr-with-tdmr-hdd, Accessed: 2019-08-23.

[8] J. Feist, *Multi Actuator Technology: A New Performance Breakthrough*, https://blog.seagate.com/craftsman-ship/multi-actuator-technology-a-new-performance-breakthrough, Accessed: 2019-08-01.

[9] *Hard Disk PNG Transparent Images*, http://www.pngall.com/hard-disk-png, Accessed: 2021-07-01.

[10] R. Merritt, *Hard drives go perpendicular*, https://www.eetimes.com/hard-drives-go-perpendicular/, Accessed: 2021-07-01.

[11] T. Feldman and G. Gibson, "Shingled magnetic recording: Areal density increase requires new data management," *The magazine of USENIX & SAGE*, vol. 38, no. 3, pp. 22–30, 2013.

[12] S. Peterson, *HAMR vs. MAMR, and the future of high-capacity hard drives*, https://searchstorage.techtarget.com/feature/HAMR-vs-MAMR-and-the-future-of-high-capacity-hard-drives, Accessed: 2021-07-01.

[13] B. Vasić and E. M. Kurtas, *Coding and signal processing for magnetic recording systems*. CRC press, 2004.

[14] E. Banan Sadeghian, "Synchronization and detection for two-dimensional magnetic recording," Ph.D. dissertation, Georgia Institute of Technology, 2016.

[15] Z.-N. Wu, J. M. Cioffi, and K. D. Fisher, "A MMSE interpolated timing recovery scheme for the magnetic recording channel," in *Proceedings of ICC'97-International Conference on Communications*, IEEE, vol. 3, 1997, pp. 1625–1629.

[16] R. D. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, and W. Schott, "A PRML system for digital magnetic recording," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 38–56, 1992.

[17] P. Kovintavewat, I. Ozgunes, E. Kurtas, J. R. Barry, and S. W. McLaughlin, "Generalized partial-response targets for perpendicular recording with jitter noise," *IEEE Transactions on Magnetics*, vol. 38, no. 5, pp. 2340–2342, 2002.

[18] G. D. Forney, "The Viterbi algorithm," *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[19] P. Chevillat, E. Eleftheriou, and D. Maiwald, "Noise-predictive partial-response equalizers and applications," in *SUPERCOMM/ICC'92 Discovering a New World of Communications*, IEEE, 1992, pp. 942–947.

[20] J. D. Coker, E. Eleftheriou, R. L. Galbraith, and W. Hirt, "Noise-predictive maximum likelihood (NPML) detection," *IEEE Transactions on Magnetics*, vol. 34, no. 1, pp. 110–117, 1998.

[21] A. Kavčić and J. M. Moura, "The Viterbi algorithm and Markov noise memory," *IEEE Transactions on Information Theory*, vol. 46, no. 1, pp. 291–301, 2000.

[22] J. Moon and J. Park, "Pattern-dependent noise prediction in signal-dependent noise," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 4, pp. 730–743, 2001.

[23] *Carnegie Mellon Settles Patent Lawsuit*, https://www.cmu.edu/news/stories/archives/2016/february/patent-lawsuit.html, Accessed: 2019-08-11.

[24] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 terabits per square inch on conventional media," *IEEE Transactions on Magnetics*, vol. 45, no. 2, pp. 917–923, 2009.

[25] S. S. Garani, L. Dolecek, J. Barry, F. Sala, and B. Vasić, "Signal processing and coding techniques for 2-D magnetic recording: An overview," *Proceedings of the IEEE*, vol. 106, no. 2, pp. 286–318, 2018.

[26] S. Nabavi and B. V. Kumar, "Two-dimensional generalized partial response equalizer for bit-patterned media," in *2007 IEEE International Conference on Communications*, Jun. 2007, pp. 6249–6254.

[27] C. K. Matcha and S. G. Srinivasa, "Target design and low complexity signal detection for two-dimensional magnetic recording," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, Dec. 2014, pp. 1–10.

[28] S. M. Khatami and B. Vasić, "Detection for two-dimensional magnetic recording systems," in *2013 International Conference on Computing, Networking and Communications (ICNC)*, IEEE, 2013, pp. 535–539.

[29] C. K. Matcha and S. G. Srinivasa, "Generalized partial response equalization and data-dependent noise predictive signal detection over media models for TDMR," *IEEE Transactions on Magnetics*, vol. 51, no. 10, pp. 1–15, 2015.

[30] J. R. Barry, B. Vasić, M. Khatami, M. Bahrami, Y. Nakamura, Y. Okamoto, and Y. Kanai, "Optimization of bit geometry and multi-reader geometry for two-dimensional magnetic recording," *IEEE Transactions on Magnetics*, vol. 52, no. 2, pp. 1–7, 2016.

[31] J. Yao, E. Hwang, B. V. Kumar, and G. Mathew, "Two-track joint detection for two-dimensional magnetic recording (TDMR)," in *2015 IEEE International Conference on Communications*, Jun. 2015, pp. 418–424.

[32] Y. Wang and B. V. Kumar, "Multi-track joint detection for shingled magnetic recording on bit patterned media with 2-D sectors," *IEEE Transactions on Magnetics*, vol. 52, no. 7, pp. 1–7, 2016.

[33] ——, "Micromagnetics-based analysis of multi-track detection with simplified 2-D write precompensation on shingled magnetic recording," *IEEE Transactions on Magnetics*, vol. 52, no. 9, pp. 1–11, 2016.

[34] B. Fan, H. K. Thapar, and P. H. Siegel, "Multihead multitrack detection for next generation magnetic recording, part I: Weighted sum subtract joint detection with ITI

estimation," *IEEE Transactions on Communications*, vol. 65, no. 4, pp. 1635–1648, 2017.

[35] ——, "Multihead multitrack detection for next generation magnetic recording, part II: Complexity reduction—Algorithms and performance analysis," *IEEE Transactions on Communications*, vol. 65, no. 4, pp. 1649–1661, 2017.

[36] B. Fan, P. H. Siegel, and H. K. Thapar, "Generalized weighted sum subtract joint detection for a class of multihead multitrack channels," *IEEE Transactions on Magnetics*, vol. 55, no. 2, pp. 1–11, 2018.

[37] E. B. Sadeghian and J. R. Barry, "The rotating-target algorithm for jointly detecting asynchronous tracks," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 9, pp. 2463–2469, 2016.

[38] C.-C. Yeh and J. R. Barry, "Approximate minimum bit-error rate equalization for binary signaling," in *Proceedings of ICC'97-International Conference on Communications*, IEEE, vol. 2, 1997, pp. 1095–1099.

[39] C.-C. Yeh, "Minimum-error-probability equalization and multiuser detection," Ph.D. dissertation, Georgia Institute of Technology, 1998.

[40] G. Forney, "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Transactions on Information Theory*, vol. 18, no. 3, pp. 363–378, 1972.

[41] J. R. Barry, E. A. Lee, and D. G. Messerschmitt, *Digital Communication*. Springer Science & Business Media, 2012.

[42] J. Riani, S. van Beneden, J. W. Bergmans, and A. H. Immink, "Near-minimum bit-error rate equalizer adaptation for PRML systems," *IEEE Transactions on Communications*, vol. 55, no. 12, pp. 2316–2327, 2007.

[43] J. Riani, "Contributions to adaptive equalization and timing recovery for optical storage systems," Ph.D. dissertation, Eindhoven University of Technology, 2008.

[44] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: Turbo-codes," in *Proceedings of ICC'93-IEEE International Conference on Communications*, IEEE, vol. 2, 1993, pp. 1064–1070.

[45] G. D. Forney, *Concatenated codes.* MIT Press, 1966.

[46] R. Gallager, "Low-density parity-check codes," *IRE Transaction on Information Theory*, vol. 8, no. 1, pp. 21–28, 1962.

[47] K. R. Narayanan, "Effect of precoding on the convergence of turbo equalization for partial response channels," *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 4, pp. 686–698, 2001.

[48] S. Ten Brink, "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Transactions on Communications*, vol. 49, no. 10, pp. 1727–1737, 2001.

[49] U. U. Fayyaz and J. R. Barry, "Polar code design for intersymbol interference channels," in *IEEE Global Communications Conference (GLOBECOM'14)*, 2014.

[50] M. Franceschini, G. Ferrari, and R. Raheli, "EXIT chart-based design of LDPC codes for inter-symbol interference channels," in *Proceedings of IST Mobile Summit*, 2005.

[51] L. Kong, Y. L. Guan, J. Zheng, G. Han, K. Cai, and K.-S. Chan, "EXIT-chart-based LDPC code design for 2D ISI channels," *IEEE Transactions on Magnetics*, vol. 49, no. 6, pp. 2823–2826, 2013.

[52] L. L. McPheters, S. W. McLaughlin, and K. R. Narayanan, "Precoded PRML, serial concatenation, and iterative (turbo) decoding for digital magnetic recording," *IEEE transactions on magnetics*, vol. 35, no. 5, pp. 2325–2327, 1999.

[53] T. V. Souvignier, M. Oberg, P. H. Siegel, R. E. Swanson, and J. K. Wolf, "Turbo decoding for partial response channels," *IEEE Transactions on Communications*, vol. 48, no. 8, pp. 1297–1308, 2000.

[54] Y. Qin and J.-G. Zhu, "TDMR with machine learning data detection channel," *IEEE Transactions on Magnetics*, 2021.

[55] A. Sayyafan, B. J. Belzer, K. Sivakumar, J. Shen, K. S. Chan, and A. James, "Deep neural network based media noise predictors for use in high-density magnetic recording turbo-detectors," *IEEE Transactions on Magnetics*, vol. 55, no. 12, pp. 1–6, 2019.

[56] J. Shen, A. Aboutaleb, K. Sivakumar, B. J. Belzer, K. S. Chan, and A. James, "Deep neural network a posteriori probability detector for two-dimensional magnetic recording," *IEEE Transactions on Magnetics*, vol. 56, no. 6, pp. 1–12, 2020.

[57] J. Moon and W. Zeng, "Equalization for maximum likelihood detectors," *IEEE Transactions on Magnetics*, vol. 31, no. 2, pp. 1083–1088, 1995.

[58] P. P. Vaidyanathan, "The theory of linear prediction," *Synthesis lectures on signal processing*, vol. 2, no. 1, pp. 1–184, 2007.

[59] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *IEEE Global Telecommunications Conference (GLOBECOM'89)*, 1989, pp. 1680–1686.

[60] A. Morello and V. Mignone, "DVB-S2: The second generation standard for satellite broad-band services," *Proceedings of the IEEE*, vol. 94, no. 1, pp. 210–227, 2006.

[61] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, no. 4, pp. 115–133, 1943.

[62] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, Elsevier, 1992, pp. 65–93.

[63] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[64] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[65] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of control, signals and systems*, vol. 2, no. 4, pp. 303–314, 1989.

[66] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[67] K. Fukushima and S. Miyake, "Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition," in *Competition and cooperation in neural nets*, Springer, 1982, pp. 267–285.

[68] Y. LeCun *et al.*, "Generalization and network design strategies," *Connectionism in perspective*, vol. 19, pp. 143–155, 1989.

[69] Y. LeCun, P. Haffner, L. Bottou, and Y. Bengio, "Object recognition with gradient-based learning," in *Shape, contour and grouping in computer vision*, Springer, 1999, pp. 319–345.

[70] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[71] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[72] C. Olah, *Understanding LSTM Networks*, https://http://colah.github.io/posts/2015-08-Understanding-LSTMs/, Accessed: 2021-08-10.

[73] A. Ashikhmin, G. Kramer, and S. ten Brink, "Code rate and the area under extrinsic information transfer curves," in *Proceedings IEEE International Symposium on Information Theory,*, IEEE, 2002, p. 115.

[74] ——, "Extrinsic information transfer functions: Model and erasure channel properties," *IEEE Transactions on Information Theory*, vol. 50, no. 11, pp. 2657–2673, 2004.

[75] S. Ten Brink, G. Kramer, and A. Ashikhmin, "Design of low-density parity-check codes for modulation and detection," *IEEE Transactions on Communications*, vol. 52, no. 4, pp. 670–678, 2004.

[76] M. W. Marcellin and H. J. Weber, "Two-dimensional modulation codes," *IEEE Journal on Selected Areas in Communications*, vol. 10, no. 1, pp. 254–266, 1992.

[77] K. Ciesielski, "On Stefan Banach and some of his results," *Banach Journal of Mathematical Analysis*, vol. 1, no. 1, pp. 1–10, 2007.

VITA

Shanwei Shi was born in 1992 and grew up in Huaian, Jiangsu Province, China. He received a B.S. degree in information engineering from Beijing Institute of Technology, Beijing, China, in 2015. In the same year, He was admitted by Georgia Institute of Technology, Atlanta, GA, USA and began to pursue a Ph.D. degree in Electrical and Computer Engineering. He has worked on several projects on magnetic recording sponsored by ASRC and IDEMA. He did an internship in Seagate Technology in 2017.