OPTIMAL PRECOMPENSATION IN HIGH-DENSITY MAGNETIC RECORDING

LIM YU CHIN, FABIAN

NATIONAL UNIVERSITY OF SINGAPORE

2006

OPTIMAL PRECOMPENSATION IN HIGH-DENSITY MAGNETIC RECORDING

LIM YU CHIN, FABIAN

(B. Eng. (Hons.), NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2006

Acknowledgements

I would like to express my deepest gratitude to my supervisors, namely Dr. George Mathew, Dr. Chan Kheong Sann and Dr. Lin Yu, Maria. I would like to thank Dr. George Mathew for all the technical discussions on my work, and grounding in the fundementals of signal processing. Other than being a great teacher, he has also been a great administrator. I am grateful for all his effort to facilitate my academic studies. I would also like to thank Dr. Chan Kheong Sann for the numerous discussions, which has helped me see things in new perspectives. Lastly, I would like to thank Dr. Lin Yu, Maria for the enriching past year, in which she has shared her expertise on specific areas in signal processing.

I would also like to express my outmost gratitude to John L. Loeb Professor Aleksandar Kavčić of Harvard University. Professor Kavčić has magnanimously invited me to visit his school during the course of this work, during which he has shared with me his wealth of experience and knowledge. He has also provided me an avenue to present my work at Information Storage Industry Consortium (INSIC) meetings, which allowed me to interact with researchers in a foreign environment. He has always treated me like his own student, and has played a key role in the preparation of this manuscript.

My warmest appreciation goes out to all my friends and collegues at the Data Storage Institute. I would especially like to thank Mr. Ashwin Kumar, Mr. Hongming Yang, Mr. An He and Ms. Kui Cai for their encouragement, support and technical advice during my study.

On a personal note, I would like to thank my family and friends, for supporting me throughout my post-graduate studies. This work would not have been possible without them.

Contents

Summary

In high-density magnetic recording, the readback signal is corrupted by signal-dependent nonlinearities. To characterize the readback signal by simple models, we can precompensate for the nonlinearities during the write process. While there exist many precompensation schemes in the literature, the optimal scheme with respect to any optimality criterion is unknown. In this work, we seek such a solution. The work in this thesis focuses on longitudinal magnetic recording, and considers two predominant nonlinearities, namely nonlinear transition shift (NLTS) and partial erasure (PE). We start with the well known mean-square error (MSE) optimality criterion, where the error is between the nonlinear signal and a desired signal. We want to obtain precompensation values which minimize the total MSE incurred by all written bits in a data sector.

The formulated MSE criterion can be viewed as a sum of individual MSE contributions by each data bit. This critical observation motivated the proposal of a dynamic programming approach. There are two main results in this thesis. The first result relies on the following simplification: the nonlinear channel characteristics can be assumed to be known. We discuss three different dynamic programming techniques to compute the precompensation values which are optimal under various conditions. The *finite-horizon* dynamic programming technique optimizes precompensation values for a finite number of data bits in a sector. Application of this said technique results in an individual optimal precompensation value for each bit, which varies with the position of the bit in a data sector. We then go on

to propose to view the number of data bits in each data sector as infinite. Doing so would allow application of an *infinite-horizon* dynamic programming technique, whereby the corresponding optimal solution does not have strict dependence on time. This brings about reduction in the complexity of the solution, which we consider to be pleasing from an implementation point of view. We consider two types of infinite-horizon dynamic programming techniques, namely, the discounted-cost technique and the average-cost technique.

The dynamic programming techniques do not explicitly give the precompensation values; they have to be extracted. The extraction procedures may be simplified by employing intuitive ideas, at the cost of optimality. We studied the performance of optimal and suboptimal methods using computer simulations. Under reasonable assumptions, the suboptimal solution is found to perform as good as the optimal solution.

The second result deals with a more complicated problem of extracting optimal precompensation when the channel characteristics are unknown. We utilize Q-learning techniques to perform this task, which require *a priori* knowledge of the NLTS. Estimation of the NLTS in the system can be done by borrowing existing NLTS measurement techniques found in the literature. We also consider incorporating equalization into our optimal precompensation algorithm. Using computer simulations, we computed optimal precompensatation for a readback signal equalized to the *extended partial response class 4* (EPR4) target. We also performed simple studies to observe the characteristics of the noise in a precompensated signal. Finally, we conclude with some comments for further work.

List of Symbols and Abbreviations

$\mathbf{A}_1^n$	notation for vector $[A_1, A_2, ..., A_n]$
A^*	optimum value of A
α	discounting factor for the discounted-cost technique
b_n	signed transition sequence
c_n	precompensation value sequence
$\mathcal{C}$	mean-square error (MSE) cost function
δ	intersymbol interference extension length
D	distance between write current transition and past written transition
Δ_n	nonlinear transition shift (NLTS) sequence
e_n	error between readback signal and some desired signal
ϵ_n	output of finite-state machine (FSM) at time n
$E\{B\}$	expected value of random variable B
γ_n	partial erasure (PE) signal attenuation sequence
$G(i)$	function used to define the average-cost technique Bellman's equation
$h(t)$	transition response
i, j, l	integer values representing states or iteration counts
I_1, I_2	anti-causal and causal intersymbol interference lengths
$J_n(i)$	cost-to-go function of state i at time n
k, n	discrete-time indices

K_1, K_2	NLTS functional form parameters
λ	average-cost (per bit)
L	length of past neighborhood of bits which affect NLTS
μ_n	dynamic programming policy at time n
N	number of bits in a data sector
P	arbitrary time index
$\Pr\{A\}$	probability of event A
x_n	written transition position sequence
$Q_n(i, \mu)$	Q-value function of state i and policy μ at time n
ρ_n	Q-learning step-size sequence
σ_v^2	variance of AWGN sample v_n
σ_m^2	media noise variance
τ	suboptimal solution memory length
T	bit period
v_n	additive white Gaussian noise (AWGN)
$V_n(i)$	value function of state i at time n
y_n	desired target signal
z_n	sampled, nonlinear readback signal
BER	bit-error rate
DC	direct current
ECC	error-control code
FSM	finite-state machine

ISI	intersymbol interference
LTI	linear time-invariant
MSE	mean-square error
MR	magnetoresistive
NLTS	nonlinear transition shift
NRZI	non-return to zero inverse
PE	partial erasure
PRML	partial response maximum-likelihood
RLL	run-length limited

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background on Magnetic Recording

The term *magnetic recording* describes the process in which data (analog or digital) is recorded onto a magnetic medium, for the purpose of storage. The first working magnetic storage device was developed in 1898 by Danish engineer Valdemar Poulsen [30]. Poulsen's motivation for building such a device was to allow people to leave voice messages on the telephone, and served as a forerunner to the modern answering machine. This marked the beginning of a multibillion dollar industry, which perpetuates due to the insatiable appetite for data storage.

This thesis focuses on magnetic recording for *hard-drive* systems. A hard-drive, also known as a *hard-disk*, allows a computer system to store data. Hard-drives are effective devices for long-term data storage. In computer systems, hard-drives are also used for temporary data storage, for instance when most of the computer's volatile memory space has been used up, and its memory needs to be freed to perform other tasks. In the early 1950's, computer engineers began their search for such a device with *tape-drives* [30]. It turns out that there exists a crippling data

access problem with tape-drives. In tape-drives, the data is recorded on various parts of a long magnetic tape, wound on a spindle-like receptacle. Data access speeds are limited by how fast the magnetic tape could be wound or unwound to expose the data-containing portion to the playback head. The innovation of the hard-drive mitigated this problem. With its magnetic medium fashioned in the form of circular disks, data access was sped up and allowed concurrent processing of jobs. The first hard-disk, developed by International Business Machines (IBM) in the early 1950's, had a minuscule capacity of 5 megabytes [30].

Fast forward to the present, and we have cheap and available hard-drives that store over 200 gigabytes. That is at least a 50,000,000% increase in storage space over 50 years. Today, a typical user's storage needs are dictated by work and leisure. Nowadays, computer programs can be as large as in the order of hundreds of megabytes, and music files, motion-picture files, digitized pictures, etc, also require an astronomical amount of storage space. While storage demands are also addressed by other forms of storage media, for example, digital versatile disks (DVD's), the hard-disk is irreplacable in terms of speed, reliability and data capacity.

The hard-disk is primarily composed of two components, namely the read/write head and the magnetic media. The recording medium is required to be of *hard* magnetic material [4], which once magnetized, does not lose its magnetism if left on its own. Important factors to consider when choosing the magnetic material for the medium include *coercivity* and *remnance* [4, 3], which determine how large a magnetic field is required to magnetize the medium and how much magnetism it retains, respectively.

When data is written on the media, the recording medium is magnetized into patterns. The data can then be retrieved by reading these magnetization patterns. The type of magnetic recording used in hard-disks can be split into two main cate-

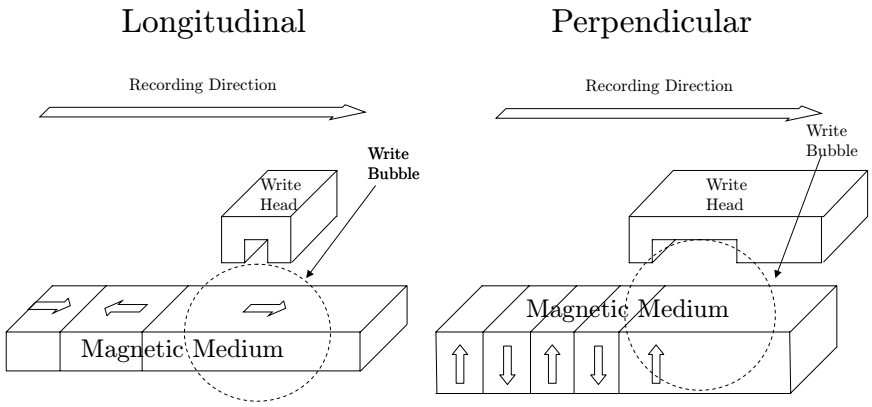


Figure 1.1: Longitudinal and perpendicular magnetic recording.

gories, namely, longitudinal and perpendicular recording. As their names suggest, they differ in the direction in which the medium is magnetized. Figure 1.1 illustrates the recording process in these two cases. The disk-like recording medium is subdivided into thin, concentric circles known as *tracks*. When data writing starts, the write head is first positioned over the desired track, and the medium is spun at very high velocities. Sophisticated head sliders prevent the write head from coming into direct contact with the medium, thus protecting the recording medium from wear. When the write head is activated, it emits a region of magnetic flux termed as the *write bubble*. This flux permeates the medium, magnetizing it in the desired direction. In digital magnetic recording systems, saturation recording is used for storing data bits. That is, there are two possible magnetization directions, each corresponding to a "0" or "1" binary digit, respectively. As shown in Figure 1.1, the medium is magnetized horizontally in longitudinal recording, and vertically in perpendicular recording. Perpendicular recording was developed as a candidate for extremely high-density recording, having a thermal decay stability advantage over longitudinal recording at very high densities [5].

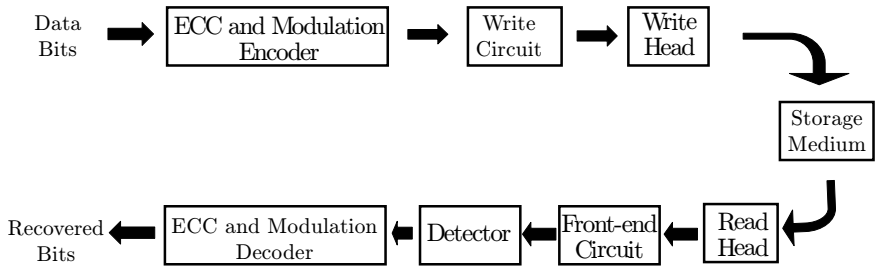


Figure 1.2: Block diagram of the magnetic recording channel.

1.2 The Magnetic Recording Channel

A hard-drive system is extremely complex, comprising many individual components designed by experts from various fields of physics and engineering. For those of us working in signal processing, we focus on a specific component known as the *recording channel*. Figure 1.2 depicts a block diagram of the recording channel, which can be modeled as a *communication system*. Specifically, the magnetic recording channel is a *baseband* communication system, where information is contained in a frequency range located around DC. In this communication system, the process of writing (or recording) the data on the medium amounts to transmission, and reading the data by means of the read head amounts to reception. The signal path from the input of the write circuit to the output of the read head constitutes the noisy transmission channel.

The error-control code (ECC) block protects the bits from detection errors by introducing redundant (non-data) symbols that provide information about the data bits. In commercial systems, powerful Reed-Solomon codes [8] are used for this purpose. A modulation code is then applied over the ECC to constrain the minimum and maximum distances between written bit transitions. The minimum distance minimizes interferences which occur when bit transitions are written too close together. The maximum distance aids sampling clock recovery by preventing a long absence of bit transitions [3]. Such modulation codes are known as *run-*

length limited (RLL) codes. A RLL code with a minimum and maximum distance symbol runlength of $d+1$ and $k+1$, respectively, is called a (d,k) RLL code. The *rate* of the RLL code is a rational number, which indicates the ratio of the number of data bits assigned to the number of coded bits. The first RLL code used was the rate 1/2 (2,7) code, which was later replaced by a rate 2/3 (1,7) code [3]. RLL codes used today do not contain the minimum distance constraint anymore, as this results in a lower code rate, which is costly when considering the high data rates in high-density magnetic recording. Code rates of current modulation codes are 8/9 [3], 16/17 [40], etc.

RLL coded bits are defined in the *non-return to zero inverse* (NRZI) convention, in which "1" and "0" binary digits indicate the presence and absence, respectively, of a bit transition. The modulation encoded bits control the write current, which in turn controls the flux switching in the write head. The write head switches flux according to the data bits to be written, magnetizing the magnetic medium. When the stored data is to be retrieved, the read head moves over the magnetized location and produces a readback signal. The readback signal is then fed into the front-end circuit, which performs noise filtering, sampling, quantization (using a analog-to-digital converter), and equalization. The equalized signal is passed to the detector which recovers the written symbols. Finally, the detected symbols pass through the modulation decoder and ECC decoder to recover the original data bits, which are then returned to the user.

1.3 Signal Processing in Magnetic Recording

The role of signal processing in magnetic recording has always been to find efficient bit detection schemes for practical implementation. This involves developing synchronization schemes, equalization schemes, detection schemes, and ECC/modulation coding schemes to improve the bit error rate (the probability that a recieved bit will be erronous). To facilitate theoretical exposition, we model

the magnetic recording channel using a mathematical model. A widely used and simple model would be the *linear time-invariant* (LTI) model. Appropriate choices for the LTI channel transition response are made from readback signal measurements. For longitudinal recording, we normally use a Lorentzian pulse, and for perpendicular recording we select from the hyperbolic tangent function [25], the inverse tangent function [29] and the Gaussian error function [42].

At low recording densities, the *peak detector* served as the primary detection scheme in longitudinal recording, which was used until the 1990's [30]. The transition response in longitudinal recording resembles a cone, with a distinct peak. The peak detector detects the locations of these peaks, which correspond to bit transitions. As recording density increased, so do the widths of the cone-shaped transition responses (relative to the symbol spacing), causing interference between neighboring written transition responses. Thus, the peaks are rendered less prominent. We term this form of interference *intersymbol interference* (ISI). Since ISI becomes larger at higher bit densities, peak detection is no longer effective at high densities.

The solution to this problem is to use equalization techniques to reduce the ISI, and to use bit detection techniques that perform well under such conditions. The well known *partial response maximum-likelihood* (PRML) detection scheme [3] falls under this category. In practical systems, we avoid using equalization techniques that remove all the ISI (known as *full response* equalization techniques), as this will result in severe noise enhancement. *Partial response* equalization techniques are used instead, whereby only some of the ISI is removed [16, 3]. After partial response equalization, we employ a Viterbi detector [3], which detects a sequence of symbols. The complexity of such a detection scheme increases exponentially as the length of the symbol sequence increases, since the number of possible symbol sequences increases exponentially with its length. Fortunately, the *Viterbi* algorithm [13, 3] mitigates this complexity problem, by providing a systematic way to eliminate

symbol sequences, thus greatly reducing the complexity.

In many instances, PRML techniques (i.e., partial response equalization followed by the Viterbi detector) are considered the *de facto* way to perform reception. The Viterbi detection scheme is theoretically optimal only if the readback samples are corrupted by additive white Gaussian noise (AWGN), i.e., the Viterbi detector chooses the symbol sequence that has the highest probabilistic likelihood of being transmitted; but unfortunately, equalization techniques correlate the noise, thus degrading its performance. However, it has been shown that by incorporating noise whiteners into the Viterbi algorithm, detection performance can be improved [10].

At high recording densities, the magnetization transitions are written too close that they start to *magnetically* interact with each other. This results in *nonlinear distortions* and are *different* from the previously mentioned ISI which is linear. Nonlinearities are caused by imperfections in the medium, and directly affect the shape of each *individual* written transition and its corresponding response. If the readback signal is nonlinear, then the readback signal can no longer be modeled by a LTI model. It is of paramount importance to study this nonlinear phenomenon, and devise solutions for this nonlinear interference problem.

1.4 Nonlinearities in Magnetic Recording and Effectiveness of Write Precompensation

Major technology improvements in magnetic recording over the past two decades have resulted in increased areal densities. Thin film media have found applications in high-density magnetic recording applications [1]. As the recording density got higher, the flux emanating from the medium got weaker. Magnetoresistive (MR) playback heads, due to their superior sensitivity [2], replaced the dual-purpose (read/write) inductive heads in data reading duties. While areal densities continue to push the envelope, resistance is encountered as the readback signal be-

comes increasingly nonlinear at high recording densities. Granular thin film media are reported to be dominated by transition noise [21], and MR playback heads are found to have nonlinear transfer characteristics [32]. Old data is not erased before data writing, and the residual magnetization patterns on the track cause signal-dependent overwrite effects [27].

The nonlinearities described above give rise to a noise that is not only correlated, but also *signal-dependent*. To combat the signal dependencies in the noise, modified sequence detectors based on the Viterbi algorithm were proposed [19, 31]. Modifications can also be made to the powerful *Baum-Welch* estimation technique [17]. The modifications to the various detectors make them much more complex than their unmodified counterparts.

This thesis focuses mainly on nonlinearities found in longitudinal recording. A dominant and well known nonlinearity found in magnetic recording is termed *nonlinear transition shift* (NLTS). NLTS causes the written bit transitions to be shifted, if there exist previously written transitions within a reasonable distance. Figure 1.3 depicts a typical sitution in which NLTS affects a bit transition. Extensive studies have revealed that NLTS is caused by *demagnetizing fields* emanating from the medium [4, 27]. These demagnetizing fields interfere with the write bubble (used to magnetize the media), and cause the bit transitions to be written in unintended positions. In longitudinal recording, NLTS moves the written bit transitions against the recording direction [4, 27], whereas the shifts occur in the opposite direction in perpendicular recording [34]. This shifting of pulses interferes severely with our bit detection, since the written bit transitions will not be evenly spaced, thus violating the linearity assumption on the channel used for developing the detector.

Write precompensation is typically used as a means to deal with NLTS. The idea is to delay the transitions in the write current, to offset the "forward" shifts

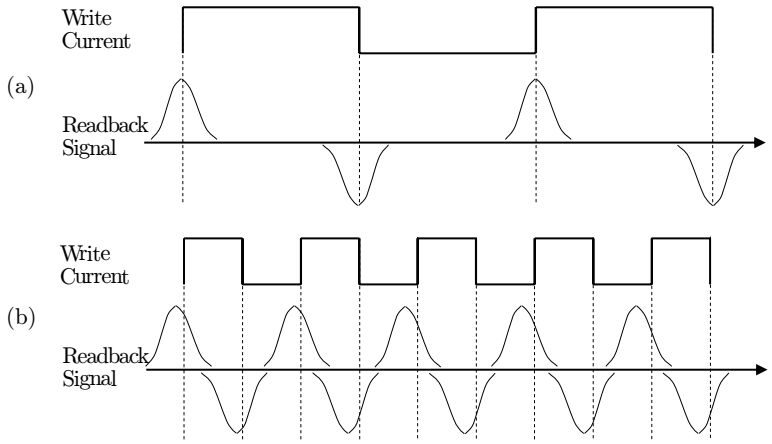


Figure 1.3: Illustration of the NLTS phenomenon. In (a), the bit transitions are written far apart and NLTS is absent. We observe alignment of the pulses with the write current transitions. In (b) the bit transitions are written closely together, and NLTS is present. We observe that the pulses occur slightly before the write current transitions.

of the written transitions caused by NLTS. Figure 1.4 shows the write precompensation subsystem of the recording system explained in the previous section. In 1987, Palmer et al. [11, 27] proposed a method to measure the NLTS in magnetic recording systems. These measurements allow the prediction of the amount of NLTS when bit transitions are written at various distances apart.

When a bit transition is to be written, we offset the write current by a certain amount determined by the NLTS measurements, such that after the NLTS occurs, the distance between any two bit transitions would be a multiple of the symbol interval (i.e., bit transitions are written at "equal" spaces apart). This technique proved effective at moderate recording densities, and sparked further research (e.g. Tsang and Tang [26, 27]) to develop different methods for NLTS measurement. IBM used write precompensation in their 1 gigabyte per square inch demonstration [15].

Today, write precompensation is found in almost every commercial hard-drive, but it is often taken for granted. In the literature, both in textbooks and research papers alike, it is obscured by "main" research topics such as bit detection,

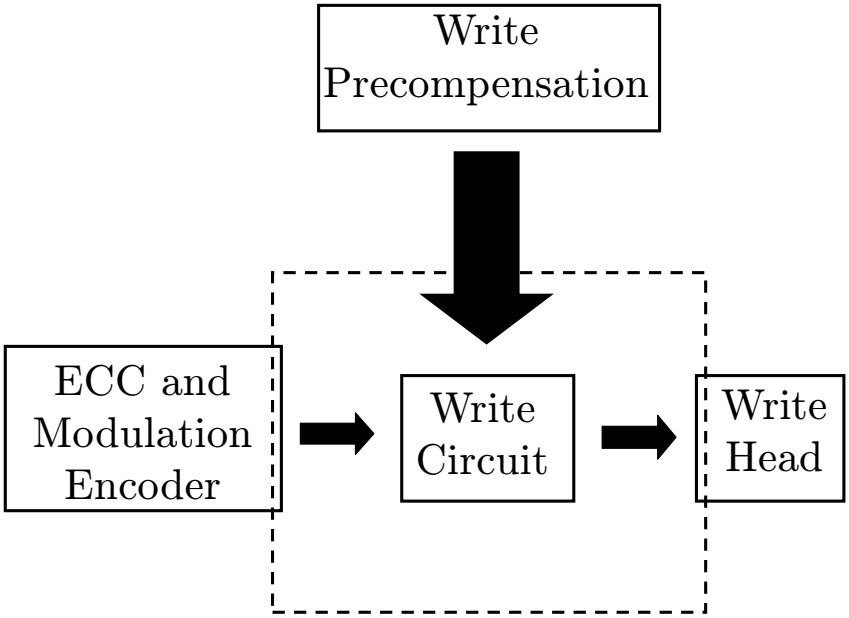


Figure 1.4: Write precompensation applied to the write current.

code designs and equalization techniques. Most hard-drive companies view their precompensation techniques as proprietary, and perhaps perform write precompensation in an ad-hoc fashion (e.g. trial-and-error approaches). Theoretically correct and/or optimum ways to approach write precompensation have not yet been reported. It is however very important that we precompensate the bits properly during the writing process, or else the readback signal will become too noisy to glean any bit information from it.

When the recording density is increased further, another nonlinearity arises. This nonlinearity is known as partial erasure (PE), which occurs when bit transitions are written very close to each other. This phenomenon is simply a percolation of magnetization domains [4], due to the close proximity in which the bit transitions are written at high recording densities. Consider a dibit, which is a pair of adjacent bit transitions. Figure 1.5 depicts PE affecting both written transitions in the dibit. The magnetized boundaries interfere with each other, causing the magnetized portion of the track (sandwiched between the two transition boundaries) to break down, causing what looks like little "islands" of magnetized media.

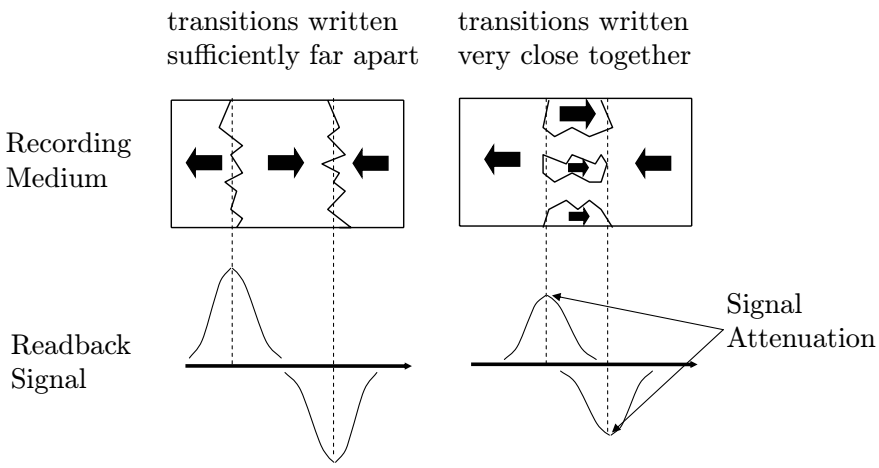


Figure 1.5: Illustration of the PE effect. When bit transitions are written too close together, we observe fragmentation of the sandwitched magnetized media. Generally, this results in an attenuated readback pulse, as illustrated by the figure.

This breakdown of the magnetized media typically causes the readback signal to be attenuated. The PE effect extends from the above argument to encompass all other signal patterns which have adjacent written bit transitions (e.g. tribits). Written bit transitions with adjacent bit transitions on *both* sides (tribit) will have higher signal attenuation due to PE than written bit transitions with adjacent bit transitions on *only one* side (dibit).

While being acknowledged as a significant source of distortion, it is not clear how to precompensate for PE. The initial train of thought was to only precompensate for NLTS, in which modifications must be made to the NLTS measurement methods which are now affected by PE. Che [9] proposed to incorporate a correction factor in the NLTS measurements, which is a function of the amount of PE. Even if the NLTS measurements are accurate, the question still remains how should we precompensate jointly for NLTS and PE. Recall from the previous section that if only NLTS is present, then we could correct for it by ensuring that the distance between any two transitions would be a multiple of the symbol interval. However, with the inclusion of the PE effect, this may not be true. It has been proposed in [24] to offset dibit patterns to alleviate the PE.

1.5 Other Types of Nonlinearities

At current recording densities, it has been reported that NLTS and PE are the most dominant types of nonlinearities [15]. Hence, in this thesis, we focus on precompensating NLTS and PE optimally, under certain assumed conditions. However, for the sake of completeness, we mention other types of nonlinearities that are also found in longitudinal magnetic recording.

Write current: In a practical scenario, the write current does not strictly resemble a square wave, as shown in Figure 1.3. The write current incurs some delay when switching polarity, and this usually results in a small misalignment of the written transition. Further, eddy currents in the head coils will cause power losses [3].

Hysteresis: The magnetic flux is not of uniform strength throughout the write bubble. The reaction of a particular magnetic medium exposed to a magnetic field is given by its *hysteresis loop* [3, 4]. The unevenness of flux in the write bubble, results in various parts of the medium (across a bit transition boundary) to be exposed to magnetic fields of varying strength. This causes the recorded transitions on the medium to take a noticeable period to change from one magnetization direction to the other. Effort must be made to choose a head/media combination which results in fast magnetization to the new direction, such that the position of the written transitions are clearly defined. Otherwise, this lack of definition results in written bit transitions shifting about.

Hard/Easy (HE) transitions: This nonlinearity is caused by previously written magnetization patterns on the track. In a practical system, erasing past written magnetization patterns is a costly process in terms of data access speed. Hence, the new data is usually written directly over past written data. The residual magnetization patterns on the medium emit demagnetization fields that interfere with the write bubble [27, 4], causing the written transitions to shift.

Pulse broadening: Another effect, also caused by the interaction of demagnetization fields with the write bubble, is known as pulse broadening. When an "interfered" write bubble records transitions on the track, the period of change from one magnetization direction to the other is larger [27] than that of the transitions recorded by a "normal" write bubble (i.e., without any interference from demagnetization fields). The period of magnetization change of a written transition affects the "width" of the corresponding response; a "wider" transitions results in a longer (or "wider") response.

1.6 Organization and Contributions of the Thesis

We consider only two nonlinearities, NLTS and PE, since they are most pertinent at currently used recording densities. While MR head nonlinearities do exist, they could be compensated for independently [6]. We formulate a theoretical treatise of the problem, as expounded in Chapter 2. We propose a dynamic programming approach to solve our optimal precompensation problem, which is elaborated in Chapters 3-4. In Chapter 5, we support the theory with simulation results. We then go on to consider a practical problem in Chapter 6, whereby the channel characteristics are assumed to be unknown. Finally, we conclude the thesis in Chapter 7, with some comments for further work.

Chapter 2

Problem Statement and Solution Approach

We start this chapter by formally defining our precompensation problem. Next, we give the relevant motivation for adopting a *dynamic programming* approach in solving the precompensation problem. We end the chapter with a simple example of a dynamic programming problem, to give the reader an initial feel for dynamic programming.

The readback signal is to be optimized with respect to some optimality criterion, such that it appears to be as close as possible to the signal from a linear channel. The readback signal, as we know, is nonlinear, and depends on the precompensation used. We mathematically formulate such a model. We also state relevant assumptions on the nonlinearities.

2.1 The Nonlinear Readback Signal Model

In the literature, models exist that capture nonlinear effects in magnetic recording. Some examples of these models are the signal-dependent autoregressive channel model [20], the transition zig-zag model [18], the position jitter width variation model [22], the microtrack model [8] and the Volterra series expansion model [14]. The typical trade-off for a nonlinear channel model is between accuracy and speed.

We formulate the simplest channel model that facilitates our purpose. We first define four important sequences b_n, c_n, Δ_n and x_n. The written data is in the form of a signed transition sequence $b_n \in \{2, -2, 0\}$ denoting positive, negative, and no transition, respectively. We will abuse terminology and refer to b_n's as signal *bits*. Let T denote the symbol interval. The amount of NLTS affecting the nth transition is defined as $\Delta_n T$, and is precompensated by offsetting the nth write current transition by $c_n T$ from the sampling instant nT. Finally, the transition position $x_n T = \Delta_n T + c_n T$ is the net shift due to the precompensation and the NLTS effect. Here, Δ_n, c_n and x_n lie in the continuous interval $(1, -1)$ [normalized by the symbol interval]. We stick to the convention where $\Delta_n, c_n > 0$ constitutes a time advance and $\Delta_n, c_n < 0$ a time delay.

We define the vectors $\mathbf{b}_{n-L}^{n}$ and $\mathbf{x}_{n-L}^{n}$ as $\mathbf{b}_{n-L}^{n} = [b_{n-L}, b_{n-L+1}, ..., b_n]$ and $\mathbf{x}_{n-L}^{n} = [x_{n-L}, x_{n-L+1}, ..., x_n]$. Normalized NLTS is defined as

$$\Delta_n \triangleq \Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n), \tag{2.1}$$

i.e., it depends on the past L bits, the positions of the past L transitions $\mathbf{x}_{n-L}^{n-1}$ and the current precompensation c_n. If there are no transitions in the past neighborhood of L bits from time n, then the NLTS value $\Delta_n = 0$. We note that we have ignored hard and soft transition effects in this model. Here, L is a constant that depends on the head-media combination.

For a written transition $b_n \neq 0$, partial erasure (PE) is defined as the amplitude attenuation (normalized to 1)

$$\gamma_n \triangleq \gamma\left(\mathbf{b}_{n-1}^{n+1}, \mathbf{x}_{n-1}^{n+1}\right). \tag{2.2}$$

If there are no neighboring transitions to a transition at time n, the partial erasure amplitude is $\gamma_n = 1$. The amount of amplitude attenuation $\gamma_n \leq 1$ depends on the

distances of the preceding/following transitions from the transition at time n.

The continuous-time, nonlinear readback signal $z(t)$ is written as

$$z(t) = \sum_{k=0}^{N-1} b_k \gamma_k h\left(t - kT + \Delta_k T + c_k T\right) + v(t), \tag{2.3}$$

where $h(t)$ is the continuous-time transition response, the term $v(t)$ is additive white Gaussian noise (AWGN), and N is the number of bits in a data sector. After bandlimiting $z(t)$ by passing it through an ideal low-pass filter of bandwidth $1/T$, we sample the output at time $t = nT$ to obtain

$$\begin{aligned} z_n &= \sum_{k=-I_1}^{I_2} b_{n-k} \gamma_{n-k} h\left(kT + \Delta_{n-k} T + c_{n-k} T\right) + v_n \\ &= \sum_{k=-I_1}^{I_2} b_{n-k} \gamma_{n-k} h\left(kT + x_{n-k} T\right) + v_n. \end{aligned} \tag{2.4}$$

We define the variance of the discrete-time AWGN random process v_n as σ_v^2. The summation limits I_1 and I_2 are the anti-causal and causal intersymbol interference (ISI) lengths, respectively.

2.2 Mean-Squared Error (MSE) Criterion

It makes sense to precompensate the nonlinear readback signal, such that it appears as close to a linear signal model as possible. Thus, detector designs can be based on the simpler linear model. Hence, we choose the mean-squared error (MSE) optimality criterion. We define the mean-squared error signal as the square of the difference between the nonlinear and linear read-back signals, given as

$$e_n^2 \triangleq \left(z_n - \sum_{k=-I_1}^{I_2} b_{n-k} h_k \right)^2, \tag{2.5}$$

where $h_k \triangleq h(t)|_{t=kT}$. Define the precompensation sequence $\mathbf{c}_0^{N-1} = [c_0, c_1, ..., c_{N-1}]$. We define the MSE *cost function* $\mathcal{C}$, which depends on the precompensation sequence $\mathbf{c}_0^{N-1}$, as the sum of the mean-squared errors for *all* sampling instants,

$$\mathcal{C} = E\left\{\sum_{n=-I_1}^{N-1+I_2} e_n^2\right\}. \tag{2.6}$$

The write precompensation is done during the write process, and we know the bits being written. Hence, we can assume that the bit sequence $\mathbf{b}_0^{N-1}$ is known. We mentioned in Section 2.1 that the NLTS and PE nonlinearities are signal-dependent. We also realize that not all bit transitions require precompensation, for example, bit transitions that are written far apart (such that NLTS and PE nonlinearity effects are not present). Hence, the optimal precompensation scheme should be specific for a particular bit sequence $\mathbf{b}_0^{N-1}$.

Thus, we can define the precompensation optimization problem as follows. We define a *new* MSE cost function

$$\mathcal{C}' = E\left\{\sum_{n=-I_1}^{N-1+I_2} e_n^2 \middle| \mathbf{b}_0^{N-1}\right\}. \tag{2.7}$$

We find the *optimal* precompensation sequence $\mathbf{c}_0^{*N-1}$ that satisfies

$$\begin{aligned} \mathbf{c}_0^{*N-1} &= \arg\min_{\mathbf{c}_0^{N-1}} E\left\{\sum_{n=-I_1}^{N-1+I_2} e_n^2 \middle| \mathbf{b}_0^{N-1}\right\} \\ &= \arg\min_{\mathbf{c}_0^{N-1}} \mathcal{C}'. \end{aligned} \tag{2.8}$$

The expectation operation in (2.8) is conditioned on the known bit sequence $\mathbf{b}_0^{N-1}$.

2.3 Motivation

The problem of minimizing the MSE cost function $\mathcal{C}'$ given in (2.7) is a complicated one. This is because the sector size N is typically large, and the number of possibilities for the optimal solution increases exponentially with N. A brute force strategy can be adopted to find such an optimal solution, however it is definitely not efficient.

The MSE $\mathcal{C}'$ is a nonlinear function of $\mathbf{c}_0^{N-1}$ since the NLTS functional form (2.1) and PE functional form (2.2) are almost certainly nonlinear functions of c_n. If we were to use some kind of numerical search method, such as the well known gradient descent algorithm [25], we may likely get stuck in a local minimum. Furthermore, the gradient of $\mathcal{C}'$ with respect to $\mathbf{c}_0^{N-1}$ will be hard to compute, even if the nonlinearities' functional forms are known. Numerical search methods are also known to be inefficient when optimizing over a large number of variables.

We observe that the cost function $\mathcal{C}'$ consists of a sum of non-negative costs $E\{e_n^2\}$. This property lends itself nicely to apply dynamic programming techniques. The phrase "dynamic programming" is a mathematical term, where the word "programming" means a set of rules that we follow while making computations. Dynamic programming, in a nutshell, solves optimization problems whereby the cost accumulates in a sequential sum.

The pioneer of this technique, Richard Bellman, noticed that the sequential sum optimization problem can be broken up in stages, where each stage corresponds to one time step [6]. At each time step n, if the optimal strategy to account for the future time $n+1$ is known, then the optimal strategy starting from the current time instant n *must* consist of the optimal strategy starting from time $n+1$. Hence, by knowing the optimal future strategy, we can solve for the current optimal strategy. Consider an arbitrary sequential sum that can be broken up into N components[1],

[1]Here, the symbol N refers to the total number of components in the sequential sum,

each assigned according to time instants n in the range $0 \leq n \leq N-1$. We work backwards, starting from the final time instant $n = N-1$, to obtain the optimal strategy for all N preceding time steps.

We apply this systematic problem solving approach to our precompensation problem. However, we are not satisfied with the MSE cost function $\mathcal{C}'$ defined in (2.7). This is because even though it is reasonable to assume that $\mathbf{b}_0^{N-1}$ is known, the optimization problem given in (2.8) amounts to an "online" approach to write precompensation, That is, during hard-drive operation, we observe the bits $\mathbf{b}_0^{N-1}$ that are to be written, and accordingly compute the optimal precompensation vector $\mathbf{c}_0^{*N-1}$. Adopting such an approach implies that we have to ensure that our optimization procedure does not cause bottleneck in the speed at which the bits $\mathbf{b}_0^{N-1}$ are to be written. Since we optimize the cost function $\mathcal{C}'$ over a typically large sequence of precompensation values $\mathbf{c}_0^{N-1}$, it is difficult to keep up with the high operation speeds of modern hard-drives.

Therefore, we want to use dynamic programming techniques to minimize the MSE cost function $\mathcal{C}$ given in (2.6) for all possible bit sequences $\mathbf{b}_0^{N-1}$. Note that this is slightly more complicated than (2.8), in the sense that the cost function $\mathcal{C}$ is *averaged* over random bits b_n. Without further theoretical development, it is not yet possible to define the minimization of the cost function $\mathcal{C}$, with respect to all possible precompensation schemes. We note that directly minimizing $\mathcal{C}$ without conditioning on the bits $\mathbf{b}_0^{N-1}$ amounts to an "offline" approach to solve the write precompensation problem; we consider all possible bit patterns $\mathbf{b}_0^{N-1}$ that *may* occur in a data sector. Nevertheless, we can still apply dynamic programming techniques. The problem of minimizing the reformulated cost function $\mathcal{C}$ in (2.6) is a *stochastic control problem* [5]. Let us consider a simple stochastic control problem, which is given as follows.

unlike in (2.6) where N was used to define the number of data bits in a sector.

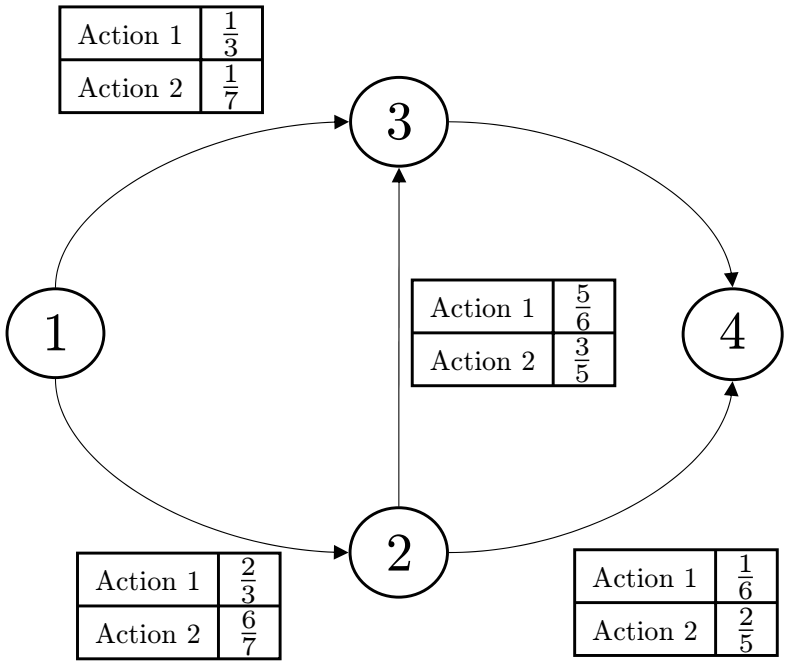


Figure 2.1: Example of a simple stochastic control problem.

Example 2.1. *Figure 2.1 depicts a simple stochastic control problem. We have 4 states, numbered from 1 to 4. Only State 1 and State 2 have two different choices for actions, each resulting in different transition probabilities. We want to determine the set of optimal actions (for States 1 and 2), such that we go from State 1 to 4 in minimum number of steps. Since we take 1 step for each transition, we assign a* cost *of 1 to all transitions. We define i as a state variable and $J^*(i)$ as the optimum cost incurred from State i.*

We work backwards to obtain the optimal solution. We start from State $i = 4$, and the optimum cost is given as

$$J^*(4) = 0. \tag{2.9}$$

This is because State 4 is the goal state, and we incur zero cost after *reaching State 4.*

Next we take a look at State $i = 3$*, and we write*

$$J^*(3) = 1. \tag{2.10}$$

This holds because at State 3, we must go to State 4 and thus incur a cost of 1 step in the process.

Next, at State $i = 2$ *we have*

$$J^*(2) = \min\left\{\underbrace{\frac{1}{6}(1 + J^*(3)) + \frac{5}{6}(1 + J^*(4))}_{\textit{Action 1}}, \underbrace{\frac{3}{5}(1 + J^*(3)) + \frac{2}{5}(1 + J^*(4))}_{\textit{Action 2}}\right\}.$$

At State 2, we have to choose between 2 actions. Action 1 allows for a transition to State 3 with probability 5/6 and a transition to the goal state (4, to be precise) with probability 1/6. Action 2 transits to State 3 with probability 3/5 and reaches the goal state with probability 2/5. Intuitively, we should choose Action 2, since it has a higher probability to reach the goal state. Let us work out the exact values to verify the intuition. Using $J^*(3) = 1$ *and* $J^*(4) = 0$*, we have*

$$\begin{aligned} J^*(2) &= \min\left\{\frac{5}{6}(2) + \frac{1}{6}(1), \frac{3}{5}(2) + \frac{2}{5}(1)\right\} \\ &= \min\left\{1\frac{5}{6}, 1\frac{3}{5}\right\} \\ &= 1\frac{3}{5}. \end{aligned} \tag{2.11}$$

As expected, Action 2 gives a lower cost.

Finally, we perform similar computations for State 1

$$\begin{aligned} J^*(1) &= \min\left\{\frac{1}{3}(1+J^*(3))+\frac{2}{3}(1+J^*(2)), \frac{1}{7}(1+J^*(3))+\frac{6}{7}(1+J^*(2))\right\} \\ &= \min\left\{\frac{1}{3}(2)+\frac{2}{3}\left(\frac{13}{5}\right), \frac{1}{7}(2)+\frac{6}{7}\left(\frac{13}{5}\right)\right\} \\ &= \min\left\{2\frac{2}{5}, 2\frac{18}{35}\right\} \\ &= 2\frac{2}{5}. \end{aligned} \tag{2.12}$$

This time Action 1 gives the minimum cost from State 1. Thus, the minimum expected number of steps to reach the goal state from State 1 is $2\frac{2}{5}$ *steps. Again, we us use intuition to verify this. We observe from Figure 2.1 that there exists a total of 3 paths to State 4, i.e.,* $1 \to 2 \to 4$, $1 \to 3 \to 4$, *and* $1 \to 2 \to 3 \to 4$. *We note that 2 of these paths require 2 steps, while the remaining path requires 3 steps. Thus, the expected number of steps should be a value between 2 and 3. We can verify that the path that requires 3 steps, under the optimal action scheme (choose Action 1 when at State 1 and choose Action 2 when at State 2), has the lowest probability of 2/5, out of all possible action scheme combinations. Hence, the expected number of steps (if we follow the opimal action scheme) will be given as*

$$\begin{aligned} 2.\Pr\{\text{2-step path is taken}\} + 3.\Pr\{\text{3-step path is taken}\} &= 2.\frac{3}{5}+3.\frac{2}{5} \\ &= 2\frac{2}{5}. \end{aligned}$$

This example is designed to give the reader an initial feel for stochastic control problems. To formulate our precompensation problem into the dynamic programming framework, a system must be defined. For any time instance n, this system resides in one of its many states. When the system transits to a new state, it generates a *one-step cost*, analogously to Example 2.1 in which each state transition

generates a cost of 1. In the next chapter (Chapter 3), we show how to formally formulate the precompensation problem into the dynamic programming framework. We then go on to discuss dynamic programming techniques for stochastic control problems that apply to our problem. In Chapter 5, we present simulation results to support the theory behind dynamic programming.

2.4 Summary

We started this chapter by defining the nonlinear readback signal model. Using the nonlinear model, we defined the MSE optimality criterion, where the error is between the nonlinear readback signal and the (desired) linear readback signal. We observed that the MSE optimality criterion is a sequential sum of MSE costs, which can be optimized using dynamic programming techniques. We proposed that the MSE cost function should be designed for "offline" computation of optimal precompensation values. Our MSE optimization problem is identified as a stochastic control problem. We gave an example of a simple stochastic control problem, and showed how it can be solved using dynamic programming techniques.

Chapter 3

Dynamic Programming

We now proceed to present three dynamic programming techniques that apply to our precompensation problem. In this chapter, the first dynamic programming technique that we present addresses the problem well, but faces a certain complexity issue. The other two techniques mitigate this complexity issue, however, they are optimal (in some sense) only if the number of data bits in a sector N approaches infinity. We advocate the use of the latter techniques as N is typically large in practice.

3.1 Finite-State Machine Model (FSM)

As noted in Section 2.3, we want to minimize the MSE cost function $\mathcal{C}$ in (2.6), over all possible precompensation schemes. We recast the precompensation problem into a form suitable for defining a finite-state machine (FSM) model through the following two steps.

1. The PE amplitude loss γ_n is dependent on the transition positions $\mathbf{x}_{n-1}^{n+1}$. Hence by utilizing equations (2.4) and (2.5), we see that e_n^2 depends on a

neighborhood of written bits $\mathbf{b}_{n-I_2-1}^{n+I_1+1}$ and transition positions $\mathbf{x}_{n-I_2-1}^{n+I_1+1}$. Hence, we express it as

$$e_n^2(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}). \tag{3.1}$$

The problem now becomes one of finding the optimal transition position sequence $\mathbf{x}_0^{*N-1}$ that minimizes the cost function $\mathcal{C}$ in (2.6). Once $\mathbf{x}_0^{*N-1}$ is obtained, we can then compute $\mathbf{c}_0^{*N-1}$ using the knowledge of the NLTS function (2.1). Notice that we have expressed the squared-error in (3.1) to be dependent on a fixed-length neighborhood of transition positions x_n. Had we chosen to write the squared-error to be dependent on precompensation values c_n, we would not be able to confine the dependence to a fixed-length window of precompensation values c_n.

2. We quantize the transition positions x_n to a finite number of values between -1 and 1.

With these modifications, we are now ready to define a FSM model.

Definition: FSM model

State: The state is defined by $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$.

Input: The input pair is $(b_{n+I_1+1}, x_{n+I_1+1})$.

a) The bit b_{n+I_1+1} is random, with transition probability $\Pr\left\{b_{n+I_1+1} \left| \mathbf{b}_{n-I_2-1}^{n+I_1} \right.\right\}$.

b) The written transition position x_{n+I_1+1} determines the *future* state $(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$.

Output: The output (one-step cost) is defined as

$$\begin{aligned}\epsilon_n &\stackrel{\Delta}{=} \epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) \\ &= E\left\{e_n^2(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) \,\middle|\, \mathbf{b}_{n-I_2-1}^{n+I_1+1}\right\}. \end{aligned} \tag{3.2}$$

Here, the expectation of the squared-error sample e_n^2 is over the noise sample v_n only, given the signal pattern $\mathbf{b}_{n-I_2-1}^{n+I_1+1}$. We see from (3.2) that the output ϵ_n depends on the present and future states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ and $(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$, respectively. Also, note that the output ϵ_n is still a random variable, as it depends on the bit sequence b_n, which is random.

The transition probability is $\Pr\left\{b_{n+I_1+1} \neq 0 \,\middle|\, \mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$, and an absence of transition occurs with probability $\Pr\left\{b_{n+I_1+1} = 0 \,\middle|\, \mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$. These values are assumed to be known *a priori*, and depend on the modulation coding applied to the signal bits. Figure 3.1 illustrates the FSM model.

Now, the MSE cost function $\mathcal{C}$ in (2.6) can be expressed as a sum of the one-step costs ϵ_n, written as

$$\begin{aligned}\mathcal{C} &= E\left\{\sum_{k=-I_1}^{N-1+I_2} e_k^2(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1})\right\} \\ &= E\left\{\sum_{k=-I_1}^{N-1+I_2} \epsilon(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1})\right\}, \end{aligned} \tag{3.3}$$

where the second equality follows from (3.2). To solve this stochastic control problem, or to minimize the MSE cost function $\mathcal{C}$, we adopt the following optimization approach. At the nth time instant and given the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$, we assign a transition position x_{n+I_1+1} to each possible bit b_{n+I_1+1}. In dynamic programming terminology [5], we term such an optimization approach as a *policy*.

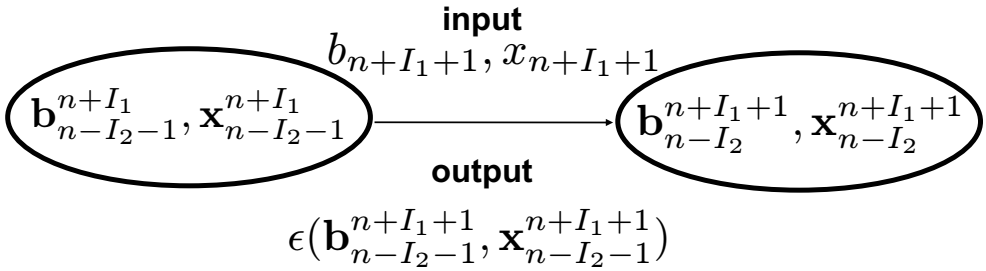


Figure 3.1: Finite-state machine (FSM) model of the precompensation problem.

3.2 Finite-Horizon Dynamic Programming

The choice of x_n is determined by the policy μ_n at time n, and we can write

$$x_{n+I_1+1} = \mu_{n+I_1+1}\left(\left(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}\right), b_{n+I_1+1}\right). \tag{3.4}$$

Thus, the policy μ_{n+I_1+1} makes a decision on the value of x_{n+I_1+1}, given the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ and the incoming bit b_{n+I_1+1}. Figure 3.2 graphically illustrates the policy μ_{n+I_1+1} for a given state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. To make visualization of this concept easier, let us make some analogies with Example 2.1. In Example 2.1, we choose an action at each state. Here, we choose an input pair assignment $(b_{n+I_1+1}, x_{n+I_1+1})$, for all possible bits b_{n+I_1+1}. In Example 2.1, the choice for the action affects the probabilities with which we transit to the new state. Here, for some bit b_{n+I_1+1}, the choice for the value $x_{n+I_1+1} = x$ indicates[2] that we go the future state $(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\big|_{x_{n+I_1+1}=x}$ with probability $\Pr\left\{b_{n+I_1+1}\,\middle|\,\mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$, and to the state $(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\big|_{x_{n+I_1+1}\neq x}$ with probability zero.

With the definition of the FSM and the policy, now we can describe the dynamic programming algorithm. The MSE optimization problem written in terms of the

[2]Here, x is a value in the range (-1,1), chosen for the sake of argument

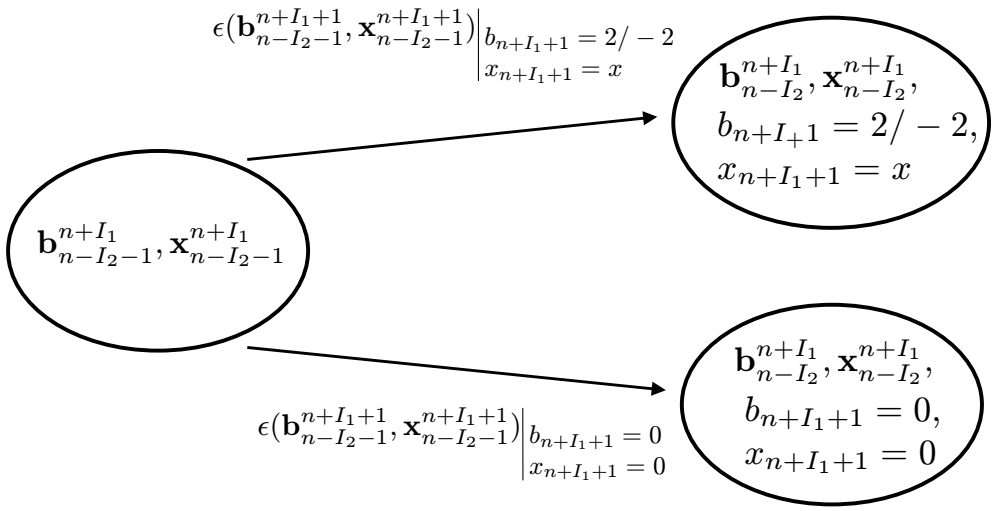


Figure 3.2: Illustration of the policy μ_{n+I_1+1} for a given state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{b}_{n-I_2-1}^{n+I_1})$. From each state, there are 2 possible transitions, corresponding to the presence or absence, respectively, of a bit written transition. When there is a transition, the bit will take on the value 2 or -2, depending on the previously written bit transition. The only exception is the all-zero state, which will have 3 possible transitions corresponding to all 3 possible values for b_n. The policy assigns corresponding values for the written transition position x_{n+I_1+1}. For the sake of illustration, x is used to indicate an arbitrary chosen transition position value. In the absence of a transition $b_{n+I_1+1}=0$, we arbitrarily set $x_{n+I_1+1}=0$.

one-step cost ϵ_n sequence and the policy μ_n sequence, is given as

$$\mathcal{C}^* \;=\; \min_{\mu_0^{N-1}} E\left\{\sum_{k=-I_1}^{N-1+I_2} \epsilon(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1})\right\}. \tag{3.5}$$

We minimize the MSE cost function $\mathcal{C}$ given in (3.3) over all policies $\mu_0^{N-1} = [\mu_0, \mu_1, ..., \mu_{N-1}]$. We assume the boundary conditions to be $(\mathbf{b}_{-\infty}^{-1}, \mathbf{x}_{-\infty}^{-1}) = (\mathbf{0}, \mathbf{0})$ and $(\mathbf{b}_N^{\infty}, \mathbf{x}_N^{\infty}) = (\mathbf{0}, \mathbf{0})$. To solve a dynamic program, recall from Section 2.3 that we work backwards from the "final" (in time) FSM state. From the assumed boundary conditions, we note that after some arbitrary policy vector μ_0^{N-1} has been chosen, we will end up in some *terminating state* denoted as $(\mathbf{b}_{N-I_1-I_2-2}^{N-1}, \mathbf{x}_{N-I_1-I_2-2}^{N-1})$. Any dynamic programming problem that terminates after a finite number of time steps is called a *finite-horizon dynamic programming* problem. Each terminating state

$(\mathbf{b}_{N-I_1-I_2-2}^{N-1}, \mathbf{x}_{N-I_1-I_2-2}^{N-1})$ is associated with its respective *terminating cost*

$$V_{N-I_1-1}^{*}(\mathbf{b}_{N-I_1-I_2-2}^{N-1}, \mathbf{x}_{N-I_1-I_2-2}^{N-1}) = E\left\{ \sum_{k=N-I_1-1}^{N-1+I_2} \epsilon(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1}) \middle| \mathbf{b}_{N-I_1-I_2-2}^{N-1} \right\}. \tag{3.6}$$

The expectation is conditioned on the signal bits $\mathbf{b}_{N-I_1-I_2-2}^{N-1}$ because the terminating cost is a function of $\mathbf{b}_{N-I_1-I_2-2}^{N-1}$. Note that the summation of one-step costs ϵ_n in (3.6) is to account for all ϵ_n that are dependent on the final bit b_{N-1}.

The dynamic programming algorithm iterates for each time instant n. Recall again from Section 2.3 that at each time instant n, we solve for the present policy, given that we know the optimal future cost. To clearly see this, first we need to define an important function. We define a *cost-to-go* function, which stores the cost accumulated from some particular state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ at time n. The *optimum* cost-to-go function [5] at time n is then written as

$$V_n^{*}(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = \min_{\mu_{n+I_1+1}^{N-1}} E\left\{ \sum_{k=n}^{N-1+I_2} \epsilon(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1}) \middle| \mathbf{b}_{n-I_2-1}^{n+I_1} \right\}. \tag{3.7}$$

The optimum cost-to-go function $V_n^{*}(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ gives the *minimum* future cost that is incurred from the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ at time n. This minimum cost is achieved using the optimum policy $\mu_{n+I_1+1}^{*N-1}$.

The finite-horizon dynamic programming algorithm is derived from the opti-

mum cost-to-go function [5] given in (3.7), and is written as

$$V_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) =$$

$$\min_{\mu_{n+I_1+1}} E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + V_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\}. \tag{3.8}$$

We make the following observations from (3.8). If the optimum future costs $V_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$ are known, we can solve for the optimal policy $\mu_{n+I_1+1}^*$. We also observe that the dynamic programming algorithm runs backwards in time, i.e., we compute the optimal costs at time n denoted by $V_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$, only after computing the optimal costs at time $n+1$ denoted by $V_{n+1}^*(\mathbf{b}_{n-I_2-1+1}^{n+I_1}, \mathbf{x}_{n-I_2-1+1}^{n+I_1})$. The recursion starts from the terminating state $(\mathbf{b}_{N-I_1-I_2-2}^{N-1}, \mathbf{x}_{N-I_1-I_2-2}^{N-1})$, and at each time instance n, we ensure the optimality of the cost-to-go function. Thus by inductive reasoning, the cost-to-go at the initial state will also be optimal, and we can obtain the *minimum* MSE.

We initialize the dynamic programming algorithm at time $n = N - I_1 - 2$, and set $V_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$ to the terminating costs $V_{N-I_1-1}^*(\mathbf{b}_{N-I_1-I_2-2}^{N-1}, \mathbf{x}_{N-I_1-I_2-2}^{N-1})$ given in (3.6). For the iteration at time n, we compute the optimal policy $\mu_{n+I_1+1}^*$ and the optimum costs $V_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ using the costs $V_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$ obtained from the previous iteration at time $n+1$. We iterate the dynamic programming algorithm until the optimal policy vector

$$\mu_0^{*N-1} = \arg\min_{\mu_0^{N-1}} E\left\{\sum_{k=-I_1}^{N-1+I_2} e_k^2(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1})\right\} \tag{3.9}$$

is obtained. Once all the iterations are done, we obtain the minimum MSE

$$\begin{aligned} \mathcal{C}^* &= \min_{\mu_0^{N-1}} E\left\{\sum_{k=-I_1}^{N-1+I_2} e_k^2(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1})\right\} \\ &= V_{-I_2-1}^*(\mathbf{b}_{-I_1-I_2-2}^{-1}, \mathbf{x}_{-I_1-I_2-2}^{-1}), \end{aligned} \tag{3.10}$$

where the *initial* state is $(\mathbf{b}_{-I_1-I_2-2}^{-1}, \mathbf{x}_{-I_1-I_2-2}^{-1})$ according to the assumed boundary conditions.

Using the finite-horizon dynamic programming algorithm, we can obtain the minimum MSE $\mathcal{C}^*$. However, the optimal policy μ_0^{*N-1} is complicated. This is because it requires that we store the policies μ_n for all states, and for all values $0 \leq n \leq N-1$. In dynamic programming terminology, this is known as a *non-stationary policy*, i.e., the optimal policy $\mu_{n+I_1+1}^*$ not only depends on the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$, but also on the time n at which the state occurs. To circumvent the complexity bottleneck arising from the non-stationary policy, in the next section, we propose to reformulate the problem into one that allows us to obtain a *stationary* optimal policy (the policies μ_n do not depend on the time instance n). The complexity reduction is obvious, and we propose to do this to efficiently solve the precompensation problem.

3.3 Infinite-Horizon Dynamic Programming

We first make an observation on the finite-horizon optimum cost-to-go function $V_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ given in (3.8). As the value of N (the sector size) gets larger, we see that the terminating costs given in (3.6) will have decreasing influence on the optimal cost $V_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. In other words, as we move further away from the boundary, the optimum future costs would be (almost) independent of the boundary conditions. Thus, decisions based on these optimal future costs will not depend on the distance from the boundaries, indirectly implying that they are independent of time. The technique of *survivor path truncation* [3], used in a Viterbi detector, is based on a very similar idea. The past optimal decisions (the point in time where the survivor paths converge) located at some reasonable distance away, would not be influenced by present optimal decisions (the point in

time where we are currently performing the add-compare-select [3]).

In magnetic recording, the sector size N is typically in the order of thousands of bits. Therefore, we propose to reformulate the problem into an *infinite-horizon stochastic control* [5] problem, which we solve using infinite-horizon dynamic programming techniques, presented in the next two sections.

3.4 Discounted-Cost Technique

Before we introduce the *discounted-cost* technique, we make a few comments. At a first glance, the discounted-cost technique may look very different from infinite-horizon dynamic programming, however, the basic concept remains the same: solve for the optimum cost (see (3.11)) that accumulates from the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. Here, the main difference is in the introduction of a "discounting factor" α, which exponentially discounts future cost incurring from the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. To see why we require this discounting factor α, let us first formulate the discounted-cost problem [5].

1. We freely choose the discounting factor α, where α must satisfy the condition $0 < \alpha < 1$.

2. Assume that all bit transition probabilities $\Pr\left\{b_{n+I_1+1} \middle| \mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$ are stationary[3].

3. Define the *discounted cost-to-go function* [5] at time n by $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$,

[3]We overlook the fact that the length of the vector $\mathbf{b}_{n-I_2-1}^{n+I_1}$ should be large for this assumption to hold. This is because a large vector $\mathbf{b}_{n-I_2-1}^{n+I_1}$ will cause the FSM state size to be huge and unmanageable.

which satisfies

$$J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = \min_{\mu_{n+I_1+1}^{\infty}} E\left\{ \sum_{k=n}^{\infty} \alpha^{k-n} \epsilon(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1}) \middle| \mathbf{b}_{n-I_2-1}^{n+I_1} \right\}. \tag{3.11}$$

Comparing (3.11) to (3.7), we note the obvious similarities and differences between the finite-horizon cost-to-go function $V_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ and the discounted cost-to-go function $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. Here, we note that we allowed the sector size N to go to infinity. The discounting factor α is required to ensure that the discounted cost-to-go function $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ is of *finite* value (the term α^{k-n} converges to zero as $k \to \infty$). Also, as shown in (3.11), the term α^{k-n} (where $k \geq n$) gets larger as the time index k gets closer to time n. This implies that the minimization is focused on one-step cost ϵ_k that occurs closer in time to the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$.

The cost-to-go function $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ gives the minimum future cost that can be incurred from the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. This minimum cost is achieved using the optimum policy $\mu_{n+I_1+1}^{*\infty}$. An important difference from the finite-horizon dynamic programming technique presented earlier, is that the optimum discounted cost-to-go function $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ is always constant for any time instance n. This can be seen by realizing that for any time instance n, the optimum future cost always depends on an infinite number of future bits. Hence, any finite time difference between two different values for n will not affect the optimal cost-to-go function $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. Of course, this holds true only if the bit probabilities $\Pr\left\{ b_{n+I_1+1} \middle| \mathbf{b}_{n-I_2-1}^{n+I_1} \right\}$ are stationary, and hence the stationarity assumption is made.

The optimal policy $\mu_{n+I_1+1}^*$ is found by solving Bellman's equation [5]. Bellman's equation is derived from the optimum cost-to-go function (3.11), and is

written as

$$J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) =$$

$$\min_{\mu_{k+I_1+1}} E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + \alpha J_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\}. \quad (3.12)$$

Since the optimal cost-to-go functions $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ are independent of the time instance n, the optimal policy $\mu_{n+I_1+1}^*$ depends only on the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. We see from (3.13) that if we know the cost $J_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$ incurred by proceeding optimally from the future state $(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$, we can solve for $\mu_{n+I_1+1}^*$ given the present state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$.

It is well known in the dynamic programming literature [5] that there are various methods available to solve Bellman's equation. We shall elaborate on one such method known as value iteration. We initialize the values $\hat{J}_0(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = 0$ for all states. Then, we update $\hat{J}_i(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ for all states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ using the value iteration equation

$$\hat{J}_i(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) =$$

$$\min_{\mu_{n+I_1+1}} E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + \alpha \hat{J}_{i-1}(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\}. \quad (3.13)$$

As $i \to \infty$, the value of $\hat{J}_i(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ will converge to $J_n^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ for all states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$, and we can obtain the optimal policy $\mu_{n+I_1+1}^*$.

3.5 Average-Cost Dynamic Programming

The discounted-cost technique presented in the previous section allows us to compute an optimal stationary policy, under the assumption that the number of bits N approaches ∞. However, an argument can be made that discounting is unnatural

for the MSE problem, where we give weights to the one-step costs appearing at all time instants n. We now present another infinite-horizon dynamic programming technique, which uses time-averaging instead to force the cost-to-go to be finite.

We denote the optimum *average* cost-to-go function[4] [5] as

$$\begin{aligned}&\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) \\ &= \min_{\mu_{n+I_1+1}^{\infty}} \lim_{N\to\infty} \frac{1}{N} E\left\{ \sum_{k=n}^{N} \epsilon(\mathbf{b}_{k-I_2-1}^{k+I_1+1}, \mathbf{x}_{k-I_2-1}^{k+I_1+1}) \middle| \mathbf{b}_{n-I_2-1}^{n+I_1} \right\}. \end{aligned} \tag{3.14}$$

The optimal average cost-to-go $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ is the optimum average-cost per time instant. Similar to the arguments presented for the discounted-cost technique, the optimum cost-to-go function $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ is independent of the time instant n since the horizon always lies infinitely far away in the future.

As in the discounted-cost technique, the average-cost technique has its own unique Bellman's equation [5], which can be solved for the optimal policy $\mu_{n+I_1+1}^*$. Define $G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ as a function of the states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. The Bellman's equation for the average-cost problem is written as

$$\begin{aligned}&G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) + \lambda^* = \\ &\qquad \min_{\mu_{n+I_1+1}} E\left\{ \epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + G^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1}) \right\}, \end{aligned} \tag{3.15}$$

and we solve it for λ^* and $G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ for all states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. The theoretical interpretation of the function $G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ is not required to understand how to solve the average-cost Bellman's equation (3.15). Since the theoretical interpretation of $G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ requires advanced concepts in dynamic programming (which is beyond the scope of this thesis), we refer the

[4] In the dynamic programming literature [5], the term $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ is often referred to as the average-cost per time instant. Since $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ can be viewed as a cost-to-go function, we avoid unnecessary introduction of new terminology.

interested reader to [5] for a discussion on the topic.

At a first glance, the Bellman's equation (3.15) for the average-cost problem may look completely unrelated to the average cost-to-go $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ given in (3.14) because we have introduced new notation λ^* and $G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$. It turns out, however, that *if* (3.15) has a solution, the optimum average cost-to-go function $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ will be independent of the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$, and will be given by λ^*. The following proposition states this formally.

Proposition 3.1. *If the following equation*

$$G^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) + \lambda^* = \min_{\mu_{n+I_1+1}} E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + G^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\}$$

is satisfied for all states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$*, then we have*

$$\lambda^* = \bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) \tag{3.16}$$

for all states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$*, where* $\bar{J}^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ *is the optimum average cost-to-go given in (3.14).*

The proof can be found in [5]. We now go on to present a simple argument to see why the optimum average cost-to-go is independent of the state. Consider two States i and j, and assume that there is a finite length path that takes us from State i to State j. Consider starting from State i, and following the path to reach State j. When we arrive at State j, we can follow the *optimal* policy that gives the optimum future cost $J^*(j)$. Since the path from State i to State j is finite in length, the cost incurred by traveling from State i to State j is negligible when the horizon approaches infinity, i.e., $N \to \infty$. Hence, the optimal average-cost incurred when starting from State i must be equal to the optimal cost incurred

when starting from State j.

Proposition 3.1 holds only if the Bellman's equation (3.15) for the average-cost problem has a solution. It can be shown that (3.15) will have a solution if there exists a state that can be reached from any other state (including itself) with non-zero probability [5]. This is stated formally by the following proposition.

Proposition 3.2. *If there exists a State l, such that any state (including itself) has a non-zero probability of reaching State l, then the Bellman's equation given in (3.15) has a solution.*

We can simply choose State l to be the all-zero state. The all-zero state exists under any reasonable choice of modulation codes, and will satisfy the conditions stated by Proposition 3.2. As with Proposition 3.1, the proof can be found in [5].

Now that we have established that the optimal policy can be obtained by solving Bellman's equation, we proceed to describe how it can be solved numerically. Though there are a couple of methods that do this [5], we describe here only the value iteration algorithm in detail. While the idea is similar to the discounted-cost technique, the average-cost value iteration algorithm is slightly more complicated. We first identify the recurrent state l, and initialize the values $\hat{G}_0(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = 0$ for all states. Next, we update $\hat{G}_0(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ for all states using the value iteration equation

$$\begin{aligned}\hat{G}_i(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = \\ \min_{\mu_{n+I_1+1}} E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + \hat{G}_{i-1}(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\} \\ - \min_{\mu_{n+I_1+1}} E\left\{\epsilon(l, l') + \hat{G}_{i-1}(l')\right\}, \end{aligned} \tag{3.17}$$

where State l' is a possible transitional state from State l, and $\epsilon(l, l')$ is defined as the one-step cost incurred by going from State $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = l$ to State

$(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1}) = l'$. It is proven [5] that the quantities $\hat{G}_i(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ will converge as $i \to \infty$, and $\min_{\mu_{n+I_1+1}} E\left\{\epsilon_n(t,t') + \hat{G}_{i-1}(t')\right\}$ will converge to the optimal average cost λ^*. We also obtain the optimal policy $\mu_{n+I_1+1}^*$, which satisfies Bellman's equation (3.15).

3.6 Summary

In this chapter, we discussed three dynamic programming techniques that can be applied to the MSE optimization problem. Before exposition of the dynamic programming details, we defined the FSM and the policy. The finite-horizon dynamic programming algorithm was described first. It computes the minimum of the MSE cost function $\mathcal{C}$, however, the corresponding optimal policy is complicated and non-stationary. To mitigate this complexity bottleneck, we proposed to use infinite-horizon dynamic programming techniques. Infinite-horizon dynamic programming assumes that the number of bits in a sector N approaches infinity, and in practice N is typically large. If the number of bits N is infinity, then the boundary conditions will be ignored, and the corresponding optimal policies will be stationary. We discussed two different types of infinite-horizon dynamic programming techniques, namely the discounted-cost technique and the average-cost technique.

Chapter 4

Extracting Precompensation Values

We have seen how to obtain the optimal policy sequence μ_0^{*N-1} (stationary or non-stationary) using the dynamic programming methods shown in Chapter 3. However, the policies themselves are not useful if we do not know how to translate them to precompensation values c_n. In this chapter, we investigate how to extract precompensation values c_n, given a sequence of optimal policies μ_n^*. We consider both optimal and suboptimal precompensation value extraction methods.

4.1 Optimal Precompensation Extraction

The optimal policy sequence μ_0^{*N-1} will give the optimal transition position sequence $\mathbf{x}_0^{*N-1}$ for any written bit sequence $\mathbf{b}_0^{N-1}$. The task now is to extract the optimal precompensation value sequence $\mathbf{c}_0^{*N-1}$ from $\mathbf{x}_0^{*N-1}$. Assuming that the NLTS functional form (2.1) is known, we can solve the equation

$$x_n^* \quad = \quad \Delta(\mathbf{b}_{n-L}^n, \mathbf{x}_{n-L}^{*n-1}, c_n) + c_n, \tag{4.1}$$

for $c_n = c_n^*$. One way of obtaining this solution is to compute the value c_n^* only after the sequence $\mathbf{x}_0^{*n}$ is obtained. The value x_n^* is a function of $\mathbf{b}_0^n$, and therefore

by (4.1), the precompensation value c_n^* is also a function of $\mathbf{b}_0^n$. Hence, since the sector size N is typically large, storing the optimal precompensation values c_n^* in a look-up table is practically infeasible.

This motivates the development of a suboptimal method to extract precompensation values from the optimal policy. In the next section, we present an intuitive approach to solving this problem. We derive a practical approximation that is easily implemented, based on a simple observation.

4.2 Suboptimal Solution

In the previous section, we argued that x_n^* depends on $\mathbf{b}_0^n$. In practice, however, we know that bits in the distant past do not really affect the present transition position x_n^*. This suggests that we can truncate the dependence on past bits to some reasonable memory length τ. We cannot prove mathematically that this property holds, but by making this assumption we greatly reduce the look-up table size. We will support our assumption with simulation results in Chapter 5.

Assumption 1: The optimal transition position x_n^* at time n is only dependent on the present signal bit b_n and τ past signal bits $\mathbf{b}_{n-\tau}^{n-1}$.

If Assumption 1 holds, it would imply that for a given signal pattern $\mathbf{b}_0^n$, the optimal transition position x_n^* would not depend on the values of bits b_k for $k < n-\tau$. Consequently, the optimal precompensation value c_n^* would depend only on the bits $\mathbf{b}_{n-\tau-L}^n$, which would significantly reduce the look-up table size.

The idea behind our suboptimal solution is therefore to limit the size of the look-up table to a length of $\tau + L + 1$ input bits. Thereby, the precompensation value $c_n(\mathbf{b}_{n-\tau-L}^n)$ is forced to be time-invariant, and can hence be written as $c_n(\mathbf{b}_{n-\tau-L}^n) = c(\mathbf{b}_{n-\tau-L}^n)$. The only remaining part is to determine the functional

form of the function $c(\cdot)$. An easy way to solve this is to set $\mathbf{b}_{-\infty}^{-1} = \mathbf{0}$ and set

$$c(\mathbf{b}_0^{\tau+L}) \stackrel{\Delta}{=} c_{\tau+L}^*(\mathbf{b}_0^{\tau+L}),$$

where $c_{\tau+L}^*(\mathbf{b}_0^{\tau+L})$ is the optimal precompensation computed in the previous section for the bit pattern $\mathbf{b}_0^{\tau+L}$.

4.3 Error Propagation

We claim that the NLTS value Δ_n depends on the precompensation sequence $\mathbf{c}_0^n$ (we will support this claim with logical reasoning later in this section). Hence, we consider the possibility of error propagation, which happens since the value Δ_n depends on $\mathbf{c}_0^n$. Let us consider that we make Assumption 1 and assume some value for τ, but the optimal precompensation sequence c_n^* actually depends on a *larger* past neighborhood of bits $\mathbf{b}_{n-\hat{\tau}-L}^n$, where $\hat{\tau} > \tau$. We want to examine what happens when we use precompensation values c_n' obtained by making Assumption 1 (suboptimal solution).

Let the sequence c_n^* denote the optimal precompensation sequence obtained using the method described in Section 4.1. Let us consider that we write the bits $\mathbf{b}_{-\infty}^{\infty}$ twice, using precompensation sequences c_n^* and c_n', respectively. Let us also assume that $c_n^* \neq c_n'$ for $n \leq P$ and $c_n^* = c_n'$ for all $n > P$, where P is an arbitrarily chosen integer for the sake of the argument that follows. Intuition suggests that if $c_n^* = c_n'$ for $n > P$, we would expect that x_n^* should be equal to x_n' for n sufficiently larger than P. Now, since $\mathbf{c}_0^{*P} \neq \mathbf{c}_0^{'P}$ and Δ_n depends on the entire vector $\mathbf{c}_0^n$ (as claimed in the previous paragraph), we conclude that $\Delta_n^* \neq \Delta_n'$, for all $n > P$. Consequently, we may get $x_n^* \neq x_n'$, for $n > P$, i.e. the error may propagate.

Now, what remains to argue is that the value Δ_n depends on $\mathbf{c}_0^n$. As seen

from equation (2.1), the value Δ_n depends on x_{n-1}. Using the relation $x_{n-1} = \Delta_{n-1} + c_{n-1}$, we see that x_{n-1} depends on Δ_{n-1} and c_{n-1}. If we continue this inductive argument, we conclude that Δ_n depends on $\mathbf{c}_0^n$.

Fortunately, in realistic scenarios, it is observed that (see Chapter 5) Δ_n depends on a short neighborhood of past precompensation values c_n. Hence, as our simulations show in Chapter 5, at time instances $n \gg P$, the transition position sequence x'_n will be equal to the optimal transition position sequence x_n^* if τ is chosen adequately large.

4.4 Summary

In this chapter, we explained how to extract optimal precompensation values from an optimal dynamic programming policy. We argued that the optimal transition position c_n^* depends on the sequence of bits $\mathbf{b}_0^n$. Thus, it is practically infeasible to store c_n^* (for all values of n) in a look-up table, since n lies in the range $0 \leq n \leq N-1$, and (the number of bits in a data sector) N is typically large. We proposed to make a practical assumption that allows us to truncate the bit dependence of x_n^* to a past bit neighborhood $\mathbf{b}_{n-\tau}^n$, where τ is a reasonable past bit memory length. This assumption in turn reduces the past bit neighborhood dependence of c_n^*, and gives us a suboptimal solution. Theoretically speaking, the suboptimal solution will suffer from error propagation. However, the error propagation is observed to be small in the simulation results shown in the next chapter.

Chapter 5

Computer Simulations

In this chapter, we test the three dynamic programming algorithms presented in Chapter 3, namely the finite-horizon dynamic programming, the discounted-cost technique, and the average-cost technique. We also test both optimal and suboptimal precompensation extraction methods presented in Chapter 4. We compare the performances of the three dynamic programming algorithms, to evaluate their effectiveness in relation to each other. The results are presented in the following manner.

1. We first show results pertaining to the discounted-cost technique when the bits are equiprobable and independent (uncoded bits). We show the MSE performance of the discounted-cost technique (Section 3.4) for various choices of α, which we also compare with the solution given by the finite-horizon technique (Section 3.2).

2. Next, we demostrate how the dynamic program works with modulation coded bits. We focus on the discounted-cost technique here, and show that the dynamic program will choose optimum strategies that are specific to the modulation code used.

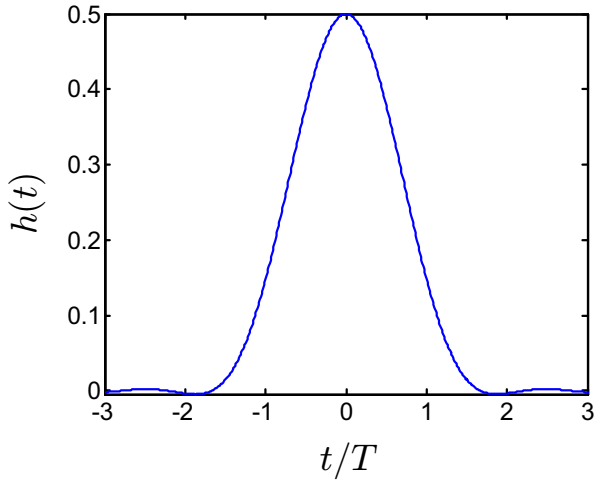


Figure 5.1: Channel transition response $h(t)$.

3. Finally, we demostrate the MSE performance of the average-cost technique (Section 3.5). We compare the MSE performance of the average-cost technique with that of the discounted-cost technique, and make some observations.

5.1 Channel Characteristics

We assume the transition response of the channel $h(t)$ is given by the following expression[5]

$$h(t) = \frac{\sin(\pi t/T')}{2\pi t/T'} \frac{\cos(2\pi t/T')}{1 - 16t^2/T'^2}, \tag{5.1}$$

where we adjusted the "pulse-width" of $h(t)$ by setting $T' = 20/9T$. The time-domain shape of $h(t)$ is shown in Figure 5.1. We mentioned in Section 2.1 that the value for the transition position x_n lies in the continuous interval $(1, -1)$. However, we can reduce the range of the interval to $-0.5 \leq x_n < 0.5$. This is because a

[5]The expression for $h(t)$ was arbitrarily chosen to resemble a plausible transition response with short ISI lengths. Though it is similar to the raised-cosine pulse used in communication channels [3], no properties of the raised-cosine pulse is of relevance here.

transition that is offset from the instant $(n+1)T$ by xT, where $0.5 < x < 1$, can also be represented[6] by a transition that is offset from the instant nT by $(1-x)T$. Hence, the closest that the bit(s) b_{n+2} and/or b_{n-2} can be written to the instant nT is $1.5T$. Referring to Figure 5.1, we observe that $h(t) \approx 0$ for values $t \leq -1.5T$ and $t \geq 1.5T$. Hence, we assume the causal and anti-causal ISI lengths to be $I_2 = 1$ and $I_1 = 1$, respectively. This is of course an unrealistically short ISI length. In real applications, the span of ISI is much larger and we need to use equalizers to reduce the ISI, of course at the expense of coloring the noise.

To model NLTS, we use a functional form similar to equation (5) in [8],

$$\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, \mathbf{c}_{n-L}^{n}) = \sum_{j=1}^{L} \frac{-b_n b_{n-j} K_1}{4\left(j + x_{n-j} - c_n\right)^{K_2}}, \tag{5.2}$$

where K_1 and K_2 are constants dependent on the physical parameters of the recording system. We set $K_1 = 0.3$ and $K_2 = 2$, which are chosen by referring to NLTS measurements given in Figure 1 in [28]. These values of K_1 and K_2 result in greater amounts of NLTS than what is shown in [28]. We set the past bit dependence length of NLTS to $L = 5$, see equation (2.1), because transitions more than 5 symbols away do not seem to contribute much to NLTS under this model. Notice that the factor $-b_n b_{n-j}/4$ evaluates to one of three possibilities: -1, 0, or $+1$.

We choose the PE function arbitrarily, however to resemble a plausible amplitude loss. If the bit transition $b_n \neq 0$ has *only one* adjacent bit transition, we set the signal attenuation factor γ_n, for both bit transitions, to the value given in Figure 5.2. Note that we require the normalized distance (either $1 + x_{n-1} - x_n$ or $1 + x_n - x_{n+1}$) to determine the exact amount of signal attenuation. If the bit $b_n \neq 0$ is flanked by 2 adjacent transitions (i.e., $b_{n-1} \neq 0$ *and* $b_{n+1} \neq 0$), then following [24], the value of γ_n is chosen as the *product* of the individual signal

[6]This argument does not apply to the first and last bits b_0 and b_{N-1}, but we ignore this slight discrepancy to conveniently reduce the range of x_n.

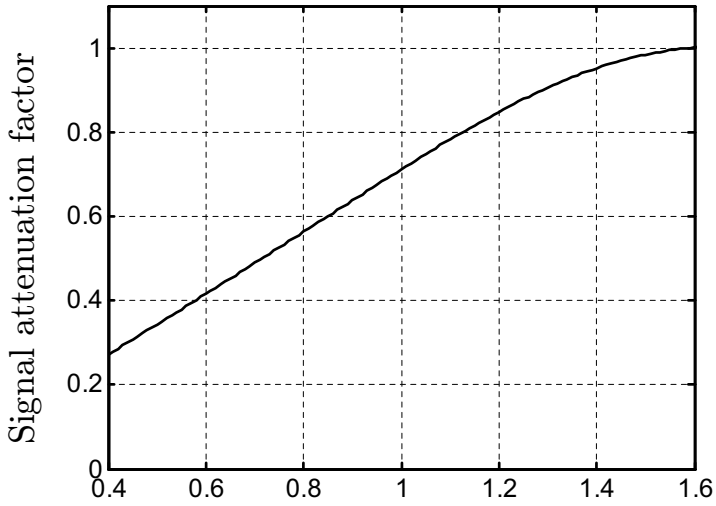
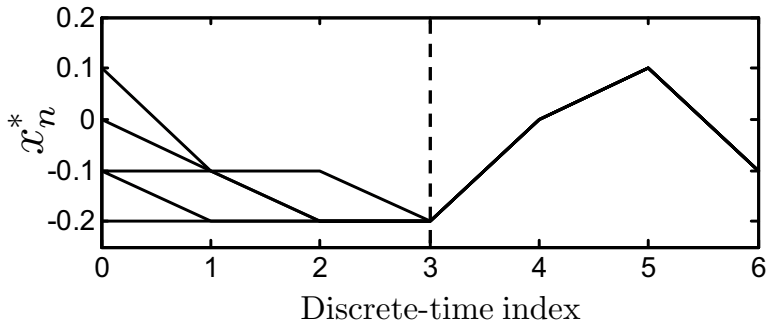


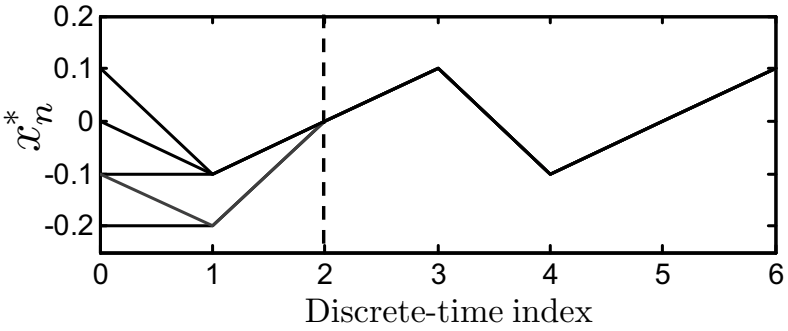
Figure 5.2: Amplitude loss resulting from partial erasure.

attenuation factors due to the bit transition pairs (b_n, b_{n-1}) and (b_n, b_{n+1}). The curve shown in Figure 5.2 is obtained using a polynomial of the 7th order, which is a function of the normalized distance between transitions. The polynomial coefficients are given in Appendix A.

We limit x_n to be quantized in intervals of 0.1, and in the range $-0.2 \leq x_n \leq 0.2$. The range of quantization $|x_n| \leq 0.2$ was chosen because of the PE function shown in Figure 5.2, as explained below. We want to ensure that PE is only caused by bits 1 symbol interval away. Therefore, the PE attenuation factor should go to 1 at the closest possible distance, so that bits 2 symbol intervals away (i.e. b_n and b_{n+2}) can be written. If we set the range $|x_n| \leq 0.2$, then the PE attenuation factor will be 1 at a distance of $(2-2(0.2))T = 1.6T$ (check with Figure 5.2). Finally, we set the bit transition probabilities $\Pr\left\{b_{n+I_1+1} \neq 0 | \mathbf{b}_{n-I_2-1}^{n+I_1}\right\} = \Pr\left\{b_{n+I_1+1} = 0 | \mathbf{b}_{n-I_2-1}^{n+I_1}\right\} = 0.5$, and the number of data bits in a sector $N = 4000$.



(a) Bit pattern $\mathbf{b}_{n-\tau}^n = \{2, -2, 2, -2, 0, 2, -2\}$



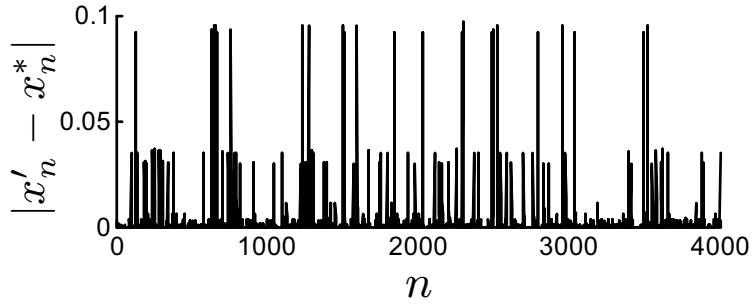
(b) Bit pattern $\mathbf{b}_{n-\tau}^n = \{2, -2, 0, 2, -2, 0, 2\}$

Figure 5.3: Optimal transition position sequences $\mathbf{x}_{n-\tau}^{*n}$ for different bit patterns $\mathbf{b}_{n-\tau}^n$. In (a), it takes 3 steps for all trajectories to converge, suggesting $\tau \geq 3$. In (b), it takes 2 steps for all trajectories to converge, suggesting $\tau \geq 2$. Taking the maximum of the two, we get $\tau \geq 3$.

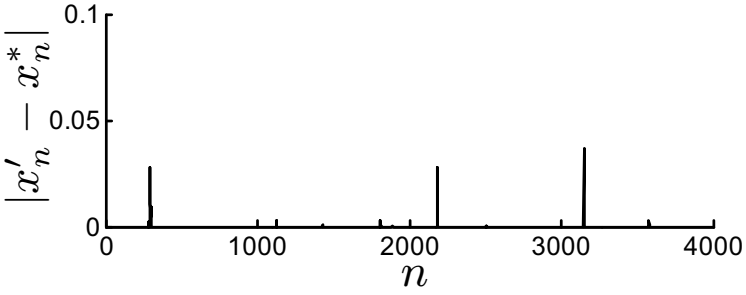
5.2 Validity of Assumption 1

The discounted-cost technique was used to compute an optimal policy. We generate a random bit sequence b_n drawn from the transition probabilities $\Pr\left\{b_{n+I_1+1}|\mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$. We then extract the optimal transition positions x_n^* using the optimal precompensation extraction method described in Section 4.1. We collect all optimal transition positions $\mathbf{x}_{n-\tau}^{*n}$ that correspond to each bit pattern $\mathbf{b}_{n-\tau}^n$, and Figure 5.3 depicts the (superimposed) trajectories. When the discounting factor is chosen to be $\alpha = 0.9$, the figure suggests that the memory of past signal bits in Assumption 1 should be chosen as $\tau \geq 4$.

Next, we test how different choices for the memory constant τ affect the error between the optimal transition position x_n^* and the suboptimal transition positions



(a) $\tau = 0$



(b) $\tau = 6$

Figure 5.4: Illustration of the effect of different values for the memory constant τ on the error between the optimal transition positions x_n^* and the suboptimal transition positions x'_n. For $\tau = 0$, the error $|x'_n - x_n^*|$ is large. For $\tau = 6$, the error $|x'_n - x_n^*|$ is practically zero, and we deem the error propagation to be negligible.

x'_n computed[7] using the method in Section 4.2. Figure 5.4 reveals that for $\tau = 6$, the error is practically zero. We also note that the isolated errors that occur for $\tau = 6$ do not seem to propagate much for time indices $k > n$. Thus, we deem the error propagation to be negligible for $\tau = 6$.

5.3 MSE Performance of the Discounted-Cost Technique

Figure 5.5 depicts the MSE performance of the suboptimal solution. We evaluate the MSE cost function $\mathcal{C}$ in (2.6) using Monte Carlo methods for different choices of the memory constant τ. We observe that for a chosen value of α, the

[7]Note that x_n^* is quantized, but x'_n is not quantized because x'_n is obtained by solving $x'_n = \Delta_n + c_n$ only after c_n is found.

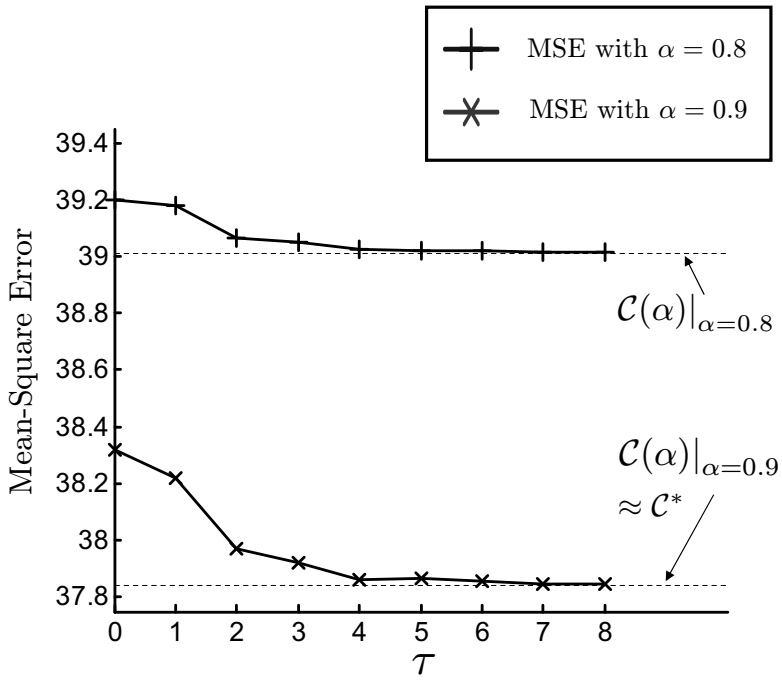


Figure 5.5: The MSE performance of the suboptimal solution, for the discounted-cost technique. The MSE approaches the optimal value $\mathcal{C}(\alpha)|_{\alpha=0.8}$ for $\tau \geq 6$. The MSE approaches the optimal value $\mathcal{C}(\alpha)|_{\alpha=0.9}$ for $\tau \geq 7$. As α approaches 1, the MSE will approach the optimal (finite-horizon) MSE value indicated by $\mathcal{C}^*$. The discounted-cost optimum $\mathcal{C}(\alpha)|_{\alpha=0.9}$ obtained is approximately the finite-horizon optimum $\mathcal{C}^*$.

MSE approaches the value $\mathcal{C}(\alpha)$, which is the MSE obtained using the optimal solution of Section 3.4. For sufficiently large τ and as α approaches 1, the MSE approaches the optimal MSE value $\mathcal{C}^*$ (as computed by finite-horizon methods). The discounted-cost optimum $\mathcal{C}(\alpha)|_{\alpha=0.9}$ approximately equals the finite-horizon optimum $\mathcal{C}^*$.

5.4 Optimum Precompensation for Coded Data Bits

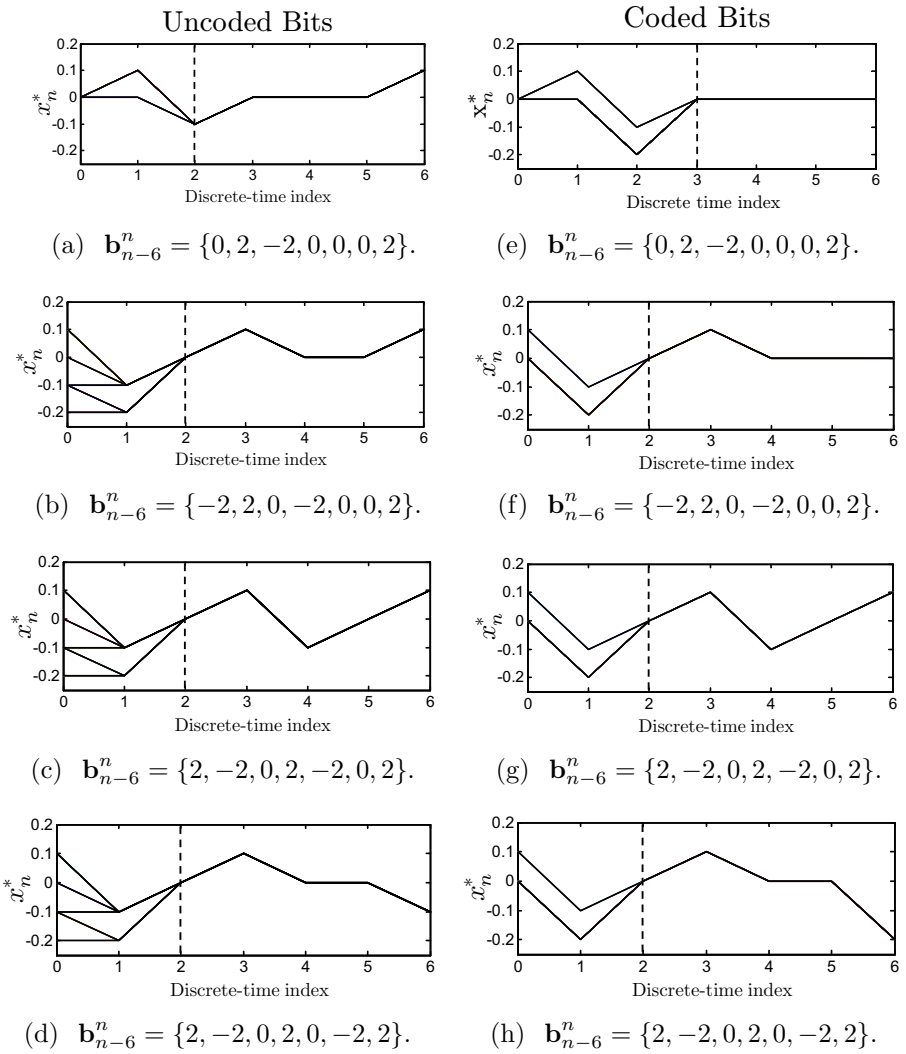
In this section, we demonstrate that our method is also able to find optimal precompensation when the data bits are coded using a modulation code. All the tests in this section are performed twice. In the first set of tests, the (uncoded) bits b_n were independently picked from an equiprobable source. The second set of tests was performed with independent, equiprobable bits coded using a modulation

code. For the sake of simplicity, we chose a rate 4/5 (0,7) maximum transition run (MTR) block code [23], where the maximum transition run is 2. The MTR code prevents tribit sequences from occurring. The bit transition probabilities $\Pr\left\{b_{n+I_1+1}|\mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$ of the coded bits are non-stationary. However, to use the discounted-cost technique, we assume stationary transition probabilities, which we estimate using simple time-averaging.

We next look at how the dynamic programming techniques optimize over different bit transition probabilities $\Pr\left\{b_{n+I_1+1}|\mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$. We set the discounting factor $\alpha = 0.9$, and picked 4 different bit patterns $\mathbf{b}_{n-6}^{n}$. Figures 5.6(a)-5.6(d) show the trajectories of the optimal transition position sequences x_n^* computed when the bits are uncoded, while Figures 5.6(e)-5.6(h) show the trajectories computed when the bits are coded. We see that the coded bits require a past bit memory length $\tau \geq 3$. Also, we see from Figure 5.6(a) that the optimal transition position x_n^* at time index 6 is 0.1, whereas Figure 5.6(e) shows its value to be 0. Similar comparisons with other figures indicate that the use of modulation code affects the optimal written transition positions x_n^*. This makes intuitive sense, as the nonlinearities are signal-dependent.

5.5 MSE for MTR Coded Data Bits

Figure 5.7 shows the MSE performance of the discounted-costs technique computed for MTR coded bits. We compare the MSE performance with that obtained when optimizing for uncoded bits. As expected of MTR codes and the precompensation process, we observe a huge improvement in the MSE performance (see Figure 5.5) when we optimize precompensation for the MTR coded bits. The MTR code prevents tribits from being written, which in turn prevents partial erasure caused by the simultaneous occurrence of preceding and following transitions. This



(a) $\mathbf{b}_{n-6}^{n} = \{0, 2, -2, 0, 0, 0, 2\}$.

(e) $\mathbf{b}_{n-6}^{n} = \{0, 2, -2, 0, 0, 0, 2\}$.

(b) $\mathbf{b}_{n-6}^{n} = \{-2, 2, 0, -2, 0, 0, 2\}$.

(f) $\mathbf{b}_{n-6}^{n} = \{-2, 2, 0, -2, 0, 0, 2\}$.

(c) $\mathbf{b}_{n-6}^{n} = \{2, -2, 0, 2, -2, 0, 2\}$.

(g) $\mathbf{b}_{n-6}^{n} = \{2, -2, 0, 2, -2, 0, 2\}$.

(d) $\mathbf{b}_{n-6}^{n} = \{2, -2, 0, 2, 0, -2, 2\}$.

(h) $\mathbf{b}_{n-6}^{n} = \{2, -2, 0, 2, 0, -2, 2\}$.

Figure 5.6: The computed optimal transition positions $\mathbf{x}_{n-6}^{*n}$ for various fixed length bit patterns $\mathbf{b}_{n-6}^{n}$. Observe that coded bits require a past bit memory length τ of at least 3. Also observe that the coded bits have different optimal transition position trajectories x_n^* as compared to the uncoded case. This is intuitively correct, since the optimization strategies depend on the bit probabilities, and nonlinearities are signal-dependent.

MSE performance gain is obtained of course, at the cost of some code rate loss.

We observe from Figure 5.7 that we practically obtain the discounted-cost optimum $\mathcal{C}(\alpha)$ for both choices of α when we set $\tau = 0$. However, we note that for $\alpha = 0.9$, Figure 5.6 has suggested the choice of $\tau \geq 3$. This is because we observe from Figure 5.6 that the optimal trajectories x_n^* do not seem to diverge much for small values of τ. Also, we observe from Figure 5.7 that we obtain a lower MSE value for $\tau = 0$, as compared to the discounted-cost optimum $\mathcal{C}(\alpha)$

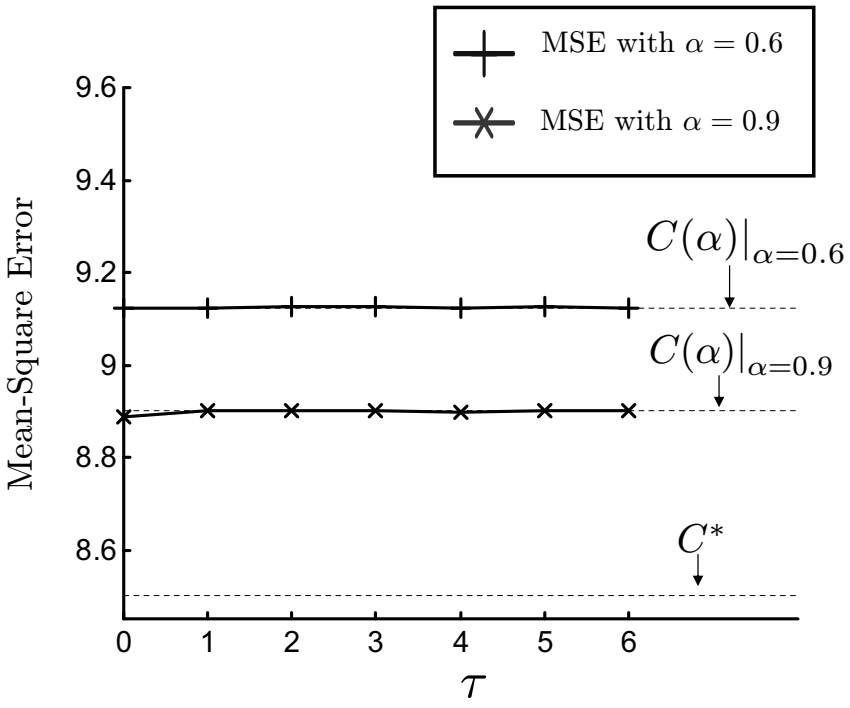
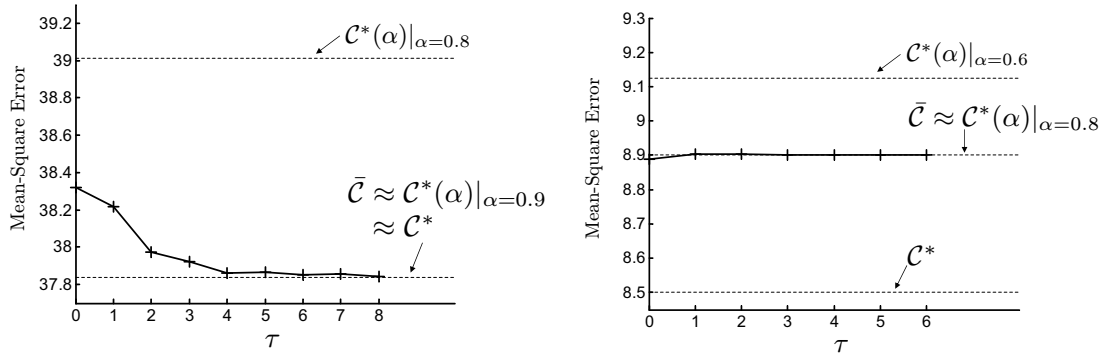


Figure 5.7: The MSE performance of the suboptimal solution when writing MTR coded bits. For values of $\alpha = 0.6$ and $\alpha = 0.9$, we get close to the discounted-cost optimum $\mathcal{C}(\alpha)$ as $\tau > 0$. As $\alpha \to 1$, the MSE approaches the minimum MSE value C^*.

for $\alpha = 0.9$. This is because we cannot guarantee that the suboptimal transition position sequence x_n will be quantized.

5.6 MSE for the Average-Cost Technique

We now present results for the average-cost technique. We show results obtained using both uncoded and coded (using the same MTR code as in Section 5.5) data bits. Figure 5.8 shows the MSE performance of the suboptimal solution for the average-cost technique. We observe that the suboptimal solution approximates the optimal solution well for all values of τ, as indicated by proximity to the average-cost optimum $\bar{\mathcal{C}}$. The discounted-cost optimum $\mathcal{C}(\alpha)$ for various choices of α are also indicated in Figure 5.8. We observe that for both uncoded bits and coded bits with $\alpha = 0.8$ and $\alpha = 0.6$, respectively, the average-cost optimal policy outperforms the discounted-cost optimal policies. Further, for both sets of data bits



(a) Uncoded bits (b) Coded bits

Figure 5.8: The MSE performance of the suboptimal solution obtained using the average cost technique. In comparison to Figures 5.5 and 5.7, we observe that the MSE performance of the average-cost optimal policy is very similar to that of the discounted-cost optimal policies (when $\alpha = 0.9$) for both uncoded bits and coded bits. We also observe that the average-cost optimal policy outperforms the discounted-cost optimal policies when we choose $\alpha = 0.6$ and $\alpha = 0.8$ for uncoded and coded bits, respectively.

with $\alpha = 0.9$ and $\alpha = 0.8$, respectively, the average-cost optimal policy performs just as well as the discounted-cost optimal policies. The optimum average-cost (see equation (3.14) and Proposition 3.1) was computed as $\lambda^* = 9.46 \times 10^{-3}$ for uncoded bits and $\lambda^* = 2.24 \times 10^{-3}$ for coded bits.

The average-cost optimum $\bar{\mathcal{C}}$ is the MSE cost function $\mathcal{C}$ (see equation (2.6)) evaluated using the average-cost optimal policy, for some N number of bits. We can write $\bar{\mathcal{C}} = \bar{\mathcal{C}}(N)$, as $\bar{\mathcal{C}}$ implicitly depends on the number of bits N. If we allow N to approach ∞, then the optimum average-cost λ^* will be equal to $\lim_{N\to\infty} \frac{1}{N}\bar{\mathcal{C}}(N)$. A quick calculation reveals that the average-cost over $N = 4000$ bits are given by 9.45×10^{-3} and 2.22×10^{-3} for uncoded and coded bits, respectively. We see that for both cases, these values are close to the theoretically computed values for infinite number of bits $\lambda^* = 9.46 \times 10^{-3}$ and $\lambda^* = 2.24 \times 10^{-3}$, respectively.

For the sake of clarity, we refrained from mentioning in Chapter 3 that we actually expect the discounted-cost optimal policy to converge to the average-cost optimal policy as $\alpha \to 1$. This is expected due to certain proven relations between the two said techniques. As seen from Figures 5.8(a) and (b), the average-

cost optimal policy obtains a MSE that is extremely close to the discounted-cost optimum $\mathcal{C}(\alpha)|_{\alpha=0.9}$, for both uncoded and coded bits. A mathematical treatise on the relation between the discounted-cost and average-cost technique can be found in [5].

5.7 Summary

In this chapter, we used computer simulations to show the performance of the various dynamic programming algorithms described in Chapter 3. In Section 5.2, using optimal transition position x_n^* trajectories computed using the discounted-cost technique, we illustrated the validity of Assumption 1 in practice. In Section 5.3, we showed that the (discounted-cost) suboptimal solution performs well with a reasonably chosen memory length τ. In Section 5.4, we showed that when the data bits are coded with a MTR code, the chosen optimum strategy is specific to the bit transition probabilities of the bits. In Section 5.5, we evaluated the MSE performance of the precompensated coded bits (with the MTR code), and observed a vast improvement in performance as compared to uncoded bits. In Sections 5.3 and 5.5, we also compared the performance of the discounted-cost technique to that of the finite-horizon technique. We observed that as the discounting factor $\alpha \to 1$, the performance of the discounted-cost technique approaches that of the finite-horizon technique. Finally, in Section 5.6, we compared the performance of the average-cost technique to the discounted-cost technique, for both suboptimal and optimal solutions.

Chapter 6

Dynamic Programming for Measured Signals

In practical situations, we do not have closed-form expressions for the one-step cost ϵ_n as a function of the present and future FSM states. This is because in practice we do not have prior knowledge of the NLTS and PE functions given in (2.1) and (2.2). The dynamic programming methods given in the previous chapters require that such knowledge is known *a priori*.

In this chapter, we describe and test (using computer simulations) dynamic programming methods that work even if the closed-form expressions for NLTS and PE are unknown. In magnetic recording, such dynamic programming methods are useful in two particular cases.

1. We want to consider a more realistic situation where equalization is used to shorten the ISI of the channel transition response. In this case, even though the FSM that generates the readback signal (prior to equalization) might be known, the FSM that generates the equalized signal is unknown.

2. We want to compute optimal precompensation given some "real" signal obtained from a spin-stand or hard-drive, where no mathematical representation of the "real" signal is known.

6.1 Q-Learning Technique

We focus on a particular type of "real" signal dynamic programming methods, known as Q-learning, and in particular, Q-learning for the discounted-cost technique. This algorithm was first proposed by Watkins [30]. As its name suggests, the algorithm tries to learn *"Quality" values*, or Q-values for short, which are denoted as

$$\begin{aligned} &Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right) \\ &\quad = E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + \alpha J^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\}. \end{aligned} \tag{6.1}$$

Recall from Section 3.3 that the quantity $J^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$ is the optimum discounted future costs and α is the discounting factor. Note that in Section 3.3, we expressed the discounted cost-to-go as $J_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$, whereas here we omit the time index $(n+1)$ in $J^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$. This is because we have previously established (in Section 3.3) that $J_{n+1}^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$ only depends on the state $(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$; we remove the dependence on the time instance n to clear up notational clutter. Note that the Q-value $Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})\right)$ inherits its time-independence from $J^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$. The Q-value equation given in (6.1) is related to Bellman's equation for the discounted-cost problem given in equation (3.13), reproduced here for the benefit of the reader:

$$\begin{aligned} &J^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = \\ &\qquad \min_{\mu_{n+I_1+1}} E\left\{\epsilon(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + \alpha J^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})\right\}. \end{aligned} \tag{6.2}$$

Note that the Q-value $Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$ is nothing but the cost-to-go function obtained without minimization over the policies μ_{n+I_1+1}. In other words, the quantity $Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$ is the cost incurred by adopting the

policy μ_{n+I_1+1} first, and then following the optimal policy which gives the optimum future cost $J^*(\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1})$. We can explicitly write

$$J^*(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}) = \min_{\mu_{n+I_1+1}} Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right). \tag{6.3}$$

Thus, the idea is to estimate these Q-values, and then perform a minimization over all policies μ_{n+I_1+1} to estimate the optimal discounted cost-to-go *and* the optimal policy. To estimate the Q-value $Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$, we utilize the following estimation update equation [30] (here $\hat{Q}^{(i)}$ denotes the estimate of Q, obtained in the ith update)

$$\begin{aligned}
&\hat{Q}^{(i)}\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right) \\
&\quad := \rho_n \left\{ e_n^2(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1}) + \alpha \min_{\mu_{n+I_1+2}} \hat{Q}^{(j)}\left((\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1}), \mu_{n+I_1+2}\right) \right\} \\
&\qquad + (1-\rho_n)\hat{Q}^{(i-1)}\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right),
\end{aligned} \tag{6.4}$$

where ρ_n is a chosen sequence of step-sizes satisfying $0 < \rho_n < 1$ for all n. To gather data for updating the Q-values, we write random data bits drawn from the known bit probabilities $\Pr\left\{b_{n+I_1+1} \middle| \mathbf{b}_{n-I_2-1}^{n+I_1}\right\}$ using an arbitrarily chosen policy μ, such that all possible states and state transitions are realized. At time instance n, we observe the state $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$, the policy μ_{n+I_1+1} that has been used, and the squared-error sample $e_n^2(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1})$. Using this information, together with some estimate of the "future" Q-value $\hat{Q}^{(j)}\left((\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1}), \mu_{n+I_1+2}\right)$, we update the "present" Q-value $\hat{Q}^{(i)}\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$ using (6.4). Note that the number of updates j, pertaining to the "future" $\hat{Q}^{(j)}\left((\mathbf{b}_{n-I_2}^{n+I_1+1}, \mathbf{x}_{n-I_2}^{n+I_1+1}), \mu_{n+I_1+2}\right)$, is arbitrary as we do not update the Q-values for *all* states $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ and policies μ_{n+I_1+1} at each time instant n. Hence, we require information about the state realizations $(\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1})$ for all time instances n. For the sake of sim-

plicity, let us for now assume that we do have this information.

As given in [12], the step-sizes are chosen as

$$\rho_n \quad = \quad 1/i^{\omega}, \tag{6.5}$$

where i is the number of updates done on the Q-value $\hat{Q}^{(i)}\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$ and ω is a real value parameter chosen in the interval (0.5,1). A lower value of ω makes the Q-value update more sensitive to new data. We also see that if we choose $\omega = 1$, we are updating the Q-value by simply averaging over all previously realized Q-values.

Q-learning is an efficient method used to learn optimal control strategies when the FSM is unknown. On a side note, an alternative solution to this problem would be to estimate the FSM first, and then run dynamic programming to get the optimal solution. However, such an approach would require us to re-run the dynamic program, should new information be gathered to update the FSM model estimate. Q-learning however, just requires updates of the Q-value update equation (6.4) to perform optimization when new information is gathered. This feature of Q-learning makes it more attractive than directly learning the FSM.

It was proven by Tsitsiklis [29] that after infinite updates of the Q-value update equation (6.4), the estimate $\hat{Q}^{(i)}\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$ will converge to $Q\left((\mathbf{b}_{n-I_2-1}^{n+I_1}, \mathbf{x}_{n-I_2-1}^{n+I_1}), \mu_{n+I_1+1}\right)$ with probability 1. Convergence is guaranteed if perfect state information is known. Unfortunately for us in data storage applications, the state information must be estimated, and hence we cannot claim convergence. However, we can resort to simulations to show "numerical" convergence.

6.2 Estimating the State Information

The FSM state comprises signal bits b_n and written transition positions x_n. Thus, knowing the FSM state is equivalent to knowing the written transition sequence x_n, since the signal bit sequence b_n is known during data writing. In practice, the transition position sequence x_n is unobservable and needs to be estimated. As in Section 2.1, we assume that the transition position sequence x_n and precompensation sequence c_n are related by the following equation

$$x_n = \Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n) + c_n,$$

where $\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n)$ is the NLTS at time n. Thus, if the NLTS function $\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n)$ is known, the written transition position x_n is known given the signal bit vector $\mathbf{b}_{n-L}^{n}$, the precompensation value c_n, and the vector of past written transition positions $\mathbf{x}_{n-L}^{n-1}$. Here, however, we assume that the NLTS function $\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n)$ is not known.

In the literature, various methods [11, 26] are proposed to estimate the NLTS function $\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n)$. We favor a method developed by Cai [7], which is able to measure the NLTS function accurately in the presence of PE. Cai's method [7] is based on the NLTS estimation method developed by Tang and Tsang [26].

To efficiently estimate the NLTS function $\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n)$, we assume that NLTS has the functional form

$$\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n) \;=\; \sum_{i=1}^{L} \frac{b_n b_{n-i}}{4} \Delta_d(D)|_{D=i+x_{n-i}-c_n}, \tag{6.6}$$

where $\Delta_d(D)$ is the NLTS observed when trying to write a bit transition at distance D from the preceding transition. In other words, the NLTS due to some signal pattern $\mathbf{b}_{n-L}^{n}$ is nothing but a linear superposition of NLTS values resulting from

the bit pairs (b_n, b_{n-i}) for $1 \leq i \leq L$. The multiplicative factor $\frac{b_n b_{n-i}}{4} \in \{-1, 0, 1\}$ accounts for the presence and polarity of the NLTS contribution from the bit pair (b_n, b_{n-i}). This superposition assumption is required because if we were to parameterize the NLTS function $\Delta(\mathbf{b}_{n-L}^{n}, \mathbf{x}_{n-L}^{n-1}, c_n)$ directly for all its variables, then such a process would be too complicated. Hence, what remains to be obtained is the function $\Delta_d(D)$ for a range of distance D. We simply quantize the distance D to a reasonable set of values within a certain range, and perform some sort of parameterization (e.g. polynomial curve fit).

6.3 Incorporating Equalization Techniques

So far, without loss of generality, we assumed that the squared-error sample is given by $e_n^2(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1})$, where the dependence on the bits b_{n-I_2-1} and b_{n+I_1+1} results from the presence of PE. Recall that the functional form of $e_n^2(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1})$ is inherited from the functional form of the readback signal $z_n(\mathbf{b}_{n-I_2-1}^{n+I_1+1}, \mathbf{x}_{n-I_2-1}^{n+I_1+1})$ (see (2.5) and (2.4)). However, when we pass the nonlinear readback signal z_n through a linear equalizer, it is not guaranteed that the functional form of the equalized readback signal will be the same as that of z_n. Hence, if we were the formulate the error signal e_n as the error between the *equalized* readback signal and the chosen *equalization target*, the functional form for e_n would be unknown. In such a case, we can just assume that the functional form of e_n^2 is approximated by

$$e_n^2(\mathbf{b}_{n-I_2-\delta}^{n+I_1+\delta}, \mathbf{x}_{n-I_2-\delta}^{n+I_1+\delta}), \tag{6.7}$$

where the limits I_1 and I_2 correspond to the ISI limits of the chosen equalization target. We term δ as the *ISI extension length*. The state is now given as $(\mathbf{b}_{n-I_2-\delta}^{n+I_1+\delta-1}, \mathbf{x}_{n-I_2-\delta}^{n+I_1+\delta-1})$. Hence, the Q-learning technique described in the earlier

part of this chapter is now applicable.

6.4 Q-Learning Simulation Results

Here, we choose the transition response $h(t)$ to be Lorentzian with a pulse width normalized density PW50/T = 3. We recall that T is the symbol interval and PW50 is the width of an isolated Lorentzian pulse at half its amplitude. We designed a (discrete-time) linear equalizer, which equalizes the discrete-time Lorentzian transition response $h_n = h(t)|_{t=nT}$ to the well-known EPR4 target. We keep the equalizer fixed for all the tests in this section, and the equalizer tap coefficients are given in Appendix B[8]. We note that a signal y_n which is equalized to the EPR4 target can be written as $y_n = b_n + 2b_{n-1} + b_{n-2}$, where the bit transition sequence $b_n \in \{2, -2, 0\}$. We run the Q-learning algorithm to compute optimal precompensation values as described previously.

Similar to the tests described in Chapter 5, we quantize x_n to 5 values, chosen in the range $-0.2 \leq x_n \leq 0.2$. Quantizing x_n to 5 discrete values turned out to be convenient when we apply Q-learning techniques, as we found that Q-learning requires a lot of data, and having 5 discrete values for x_n limited the data required to a reasonable amount. Before we apply the Q-learning algorithm, the state information has to be estimated using NLTS measurements, as explained in Section 6.2. We set the Q-learning step-size parameter (see (6.5)) as $\omega = 0.55$. We used equiprobable (uncoded) bits b_n in the tests presented in this section.

In this set of tests, we increased the amount of NLTS, as compared to the simulations performed in Chapter 5. We stick to the NLTS model given in (5.2), but we set the NLTS model parameters $K_1 = 0.4$ and $K_2 = 1.5$. The NLTS past bit

[8]We note that such a scheme, whereby one fixes the equalizer and then computes precompensation values, will not truly minimize the MSE. However, the development of joint equalization-precompensation schemes is beyond the scope of this thesis.

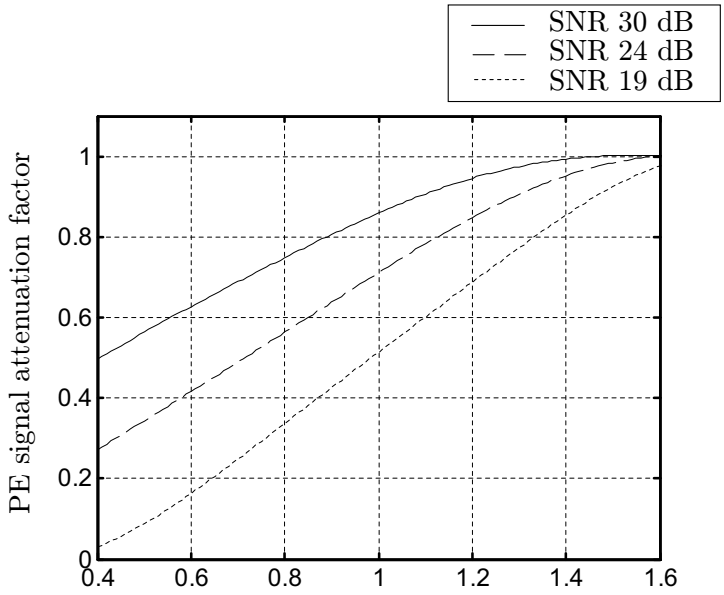


Figure 6.1: Partial erasure (PE) functions chosen for the tests.

dependence length L was chosen to be 10, for similar reasons given in Chapter 5. We note that the set of NLTS model parameters used here results in a much larger NLTS past bit dependence length (L was determined to be 5 in Chapter 5).

We carefully chose three different PE functions, shown in Figure 6.1, which represent signal attenuation behavior for three different head-media combinations. This is done to test the algorithm functionality. To keep things simple, we fixed the NLTS model for all three head-media combinations. We did so for two additional reasons. Firstly, we do not have data which describe how NLTS varies with PE. Secondly, we chose the NLTS parameters K_1 and K_2, such that they result in relatively large values for the NLTS (dibit NLTS amounts to about 40% of the bit period T). Thus, we test the algorithm under conditions that are possibly more demanding than those found in realistic scenarios. The three polynomials used to model the PE functions shown in Figure 6.1 are given in Appendix A.

We define the signal-to-noise ratio (SNR) as

$$\text{SNR} = 10 \log_{10} \left(\frac{1}{\sigma_m^2 + \sigma_v^2} \right),$$

where σ_m^2 and σ_v^2 correspond to the variances of the *media noise* (resulting from nonlinearities) and *electronic noise* (modeled by AWGN samples v_n defined in (2.4)), respectively. Given some value for the SNR, we fix the media and AWGN noise variances such that they account for 90% and 10%, respectively, of the total noise variance $\sigma_m^2 + \sigma_v^2$. The media noise variance σ_m^2 is defined as the mean-square value of the per-bit difference of the "AWGN-less" nonlinear readback signal with its linear counterpart. When computing the media noise variance σ_m^2, we write bits that are spaced at a distance of T (symbol interval) apart.

6.4.1 NLTS Measurements

As mentioned in Section 6.2, we first obtain the dibit NLTS function $\Delta_d(D)$, where D is the distance of the transition being written from the past transition (before NLTS is taken into account). Figure 6.2 shows a set of obtained NLTS measurements, for the head/media combination that results in a SNR of 19 dB (see Figure 6.1). The figure also shows a polynomial curve fit used to parameterize the function $\Delta_d(D)$, where the polynomial order was set to 6. We performed measurements for values of D chosen in the range $0.1 \leq D \leq 7.5$, ignoring NLTS effects that occur for values of $D > 7.5$.

6.4.2 Observations on Q-value Convergence

We set the ISI extension length to $\delta = 1$ (see Section 6.3). Figure 6.3 shows the optimum discounted-cost $J^*(\mathbf{b}_{n-2}^{n+1}, \mathbf{x}_{n-2}^{n+1})$ as estimated by the Q-learning algorithm.

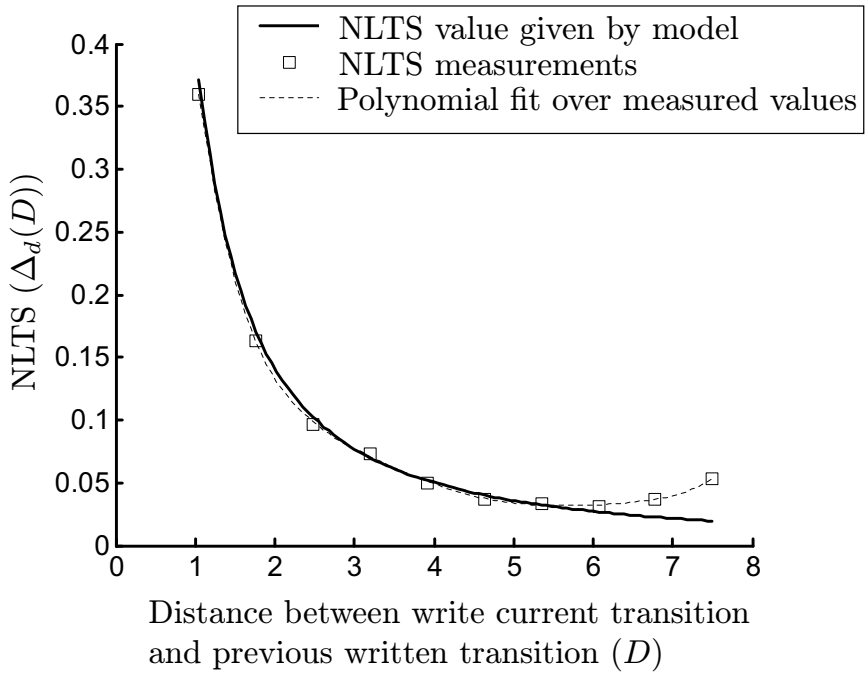


Figure 6.2: NLTS measurements obtained using Cai's method [7]. The results shown here is obtained using the PE function that results in an evaluated SNR of 19 dB (see Figure 6.1). Approximately 15 million bits were written to gather data. We observe some slight descrepencies between measurements and actual values for $D > 6.0$.

We present results for various SNR's, and the optimum future cost estimates correspond to the all-zero state $(\mathbf{b}_{n-2}^{n+1}, \mathbf{x}_{n-2}^{n+1}) = (\mathbf{0}, \mathbf{0})$. We only show optimal costs corresponding to one state, since the cost-to-go functions $J^*(\mathbf{b}_{n-2}^{n+1}, \mathbf{x}_{n-2}^{n+1})$ are interdependent.

We see that as the number of updates of the Q-values becomes large, the Q-learning algorithm converges. As a point of comparison, we used the estimated optimal policy (as computed by the Q-learning algorithm), and evaluated the discounted-cost $J^*(\mathbf{b}_{n-2}^{n+1}, \mathbf{x}_{n-2}^{n+1})$ using Monte Carlo techniques. The evaluated discounted-cost is shown using the horizontal lines in Figure 6.3. We note that the Q-learning algorithm converges pretty close to the evaluated discounted-cost, indicating that the Q-learning's estimates are fairly unbiased.

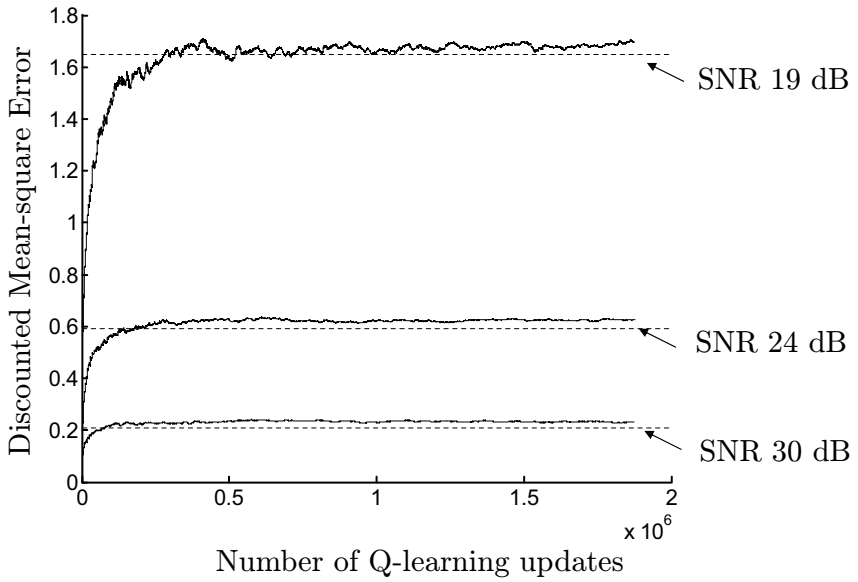


Figure 6.3: Optimum discounted-cost $J^*(\mathbf{b}_{n-2}^{n+1}, \mathbf{b}_{n-2}^{n+1})$, estimated using the Q-learning technique. Three sets of data are shown, each corresponding to the three different PE functions shown in Figure 6.1. The horizontal lines represent the discounted-cost $J^*(\mathbf{b}_{n-2}^{n+1}, \mathbf{b}_{n-2}^{n+1})$, evaluated by Monte Carlo simulations, using the optimal policy obtained from the estimated Q-values. Observe that for all three cases, as the number of updates becomes large, we approach reasonably close to the Monte Carlo simulated values.

6.4.3 Effect of ISI Extension Length δ on Optimal and Suboptimal MSE Performance

We use the Q-learning algorithm to minimize the MSE, while assuming various values for the ISI extension length δ (see Section 6.3). We also test the suboptimal solution (given in Section 4.2) for various combinations of choices for the memory constant τ and δ. We observe from Figure 6.4 that when we choose $\delta = 0$ (i.e., no ISI extension), we obtain much worse MSE performance than if we choose $\delta = 1$ or $\delta = 2$. This is because $\delta = 0$ makes the FSM state too short, and hence the nonlinear readback signal is badly represented by the FSM. Interestingly, we observe that a choice of $\delta = 1$ gives slightly better MSE performance compared to that of $\delta = 2$. Perhaps, this can be explained by noting the following. A FSM consisting of more states may result in a more detailed representation of the signal. However, a larger δ causes the state of the FSM to depend on a larger number of

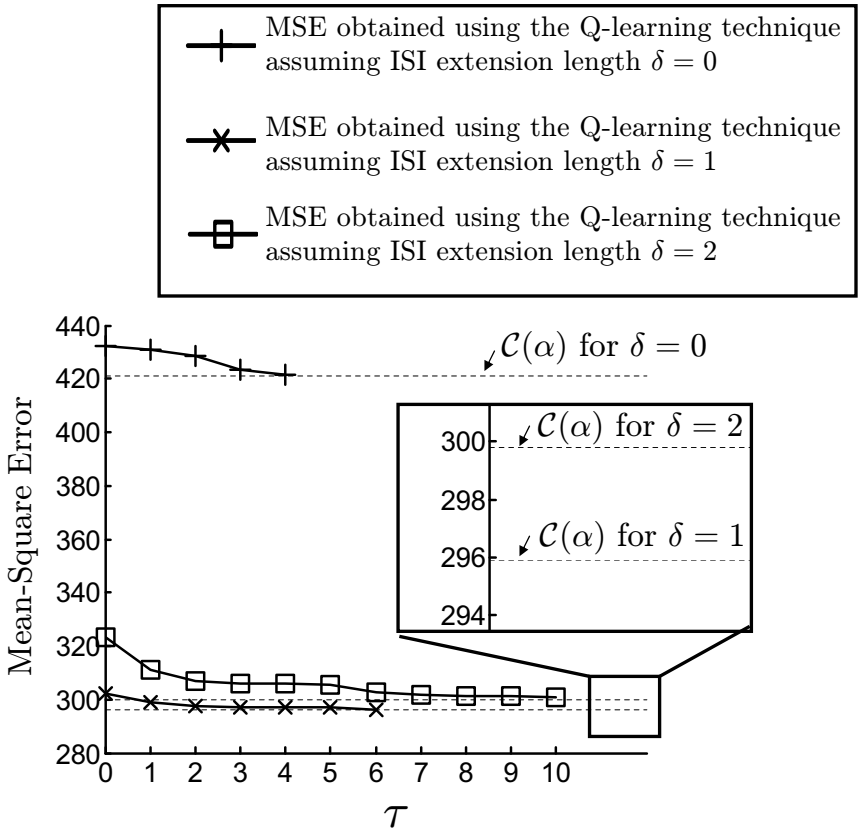


Figure 6.4: Comparison of the MSE obtained using the Q-learning technique for different choices of the ISI extension length δ. The plots indicate the MSE performance of the suboptimal solution (explained in Section 4.2), for various choices of the memory constant τ. We also include the MSE performance of the optimal solution, indicated by the horizontal, dotted lines. Observe that for a reasonable choice of τ, the MSE performance of the suboptimal solution approaches that of the optimal solution. Further, a choice of $\delta = 1$ results in a huge improvement in MSE as compared to $\delta = 0$, while choosing either $\delta = 1$ or $\delta = 2$ results in similar MSE performance.

transition position values x_n (see (6.7)). Thus, a larger value for δ causes the FSM to be more sensitive to state estimation errors, since each x_n value will suffer from estimation errors. However, we observe from Figure 6.4 that the difference in MSE performances when we choose $\delta = 1$ and $\delta = 2$ is practically small enough to be ignored. From our results, we conclude that the choice of $\delta = 1$ gives an excellent compromise between MSE performance and algorithm complexity.

6.4.4 Effect of Precompensation on Media Noise

We redefine $e_n = z_n - y_n$ (see equation (2.5)) as the error between the real

(noisy) compensated readback signal and the desired EPR4 signal $y_n = b_n + 2b_{n-1} + b_{n-2}$. This error signal represents the *compensation error*, i.e., the remnant noise in the readback waveform after performing precompensation and equalization. We set the PE function to the one that corresponds to a SNR of 24 dB. We compute optimal precompensation while setting the bit extension length $\delta = 1$. Next, we extract precompensation values using the suboptimal solution, while choosing the memory constant $\tau = 6$. We then collect all error signals e_n corresponding to each signal pattern $\mathbf{b}_{n-2}^n$. Figure 6.5 shows the histograms of the error signals e_n. Note that only 8 patterns are shown, the rest being similar as we do not consider pulse asymmetries. For the sake of comparison, we also show histograms obtained when we write bit transitions such that the distance between any two transition is a multiple of the symbol interval T. We observe from Figure 6.5 that the errors improve after applying optimal precompensation. We observe that most signal patterns, due to the nonlinearities, have error histograms with multiple "peaks". We see that optimal precompensation helps to reduce these "peaks", leaving only one main "peak" located reasonably close to zero, and making the error more "Gaussian-like". We observe that while our algorithm works well for most signal patterns, it seems to have trouble with the bit pattern $\mathbf{b}_{n-2}^n = \{-2, 2, 0\}$. It seems that this particular signal pattern may only be handled using a look-ahead policy, which we briefly expose in the next section.

6.4.5 Comparison with Look-Ahead Policies

In this subsection, we would like to point out a subtle limitation of our method. We compare the MSE performance obtained with our optimal method to a look-ahead policy, such as the one given below. We consider a simple precompensation strategy: write all dibits further apart to alleviate PE. This translates to a look-ahead policy because when deciding to shift a particular transition, we need to look one bit into the future to know if the said bit belongs to a dibit pattern. The

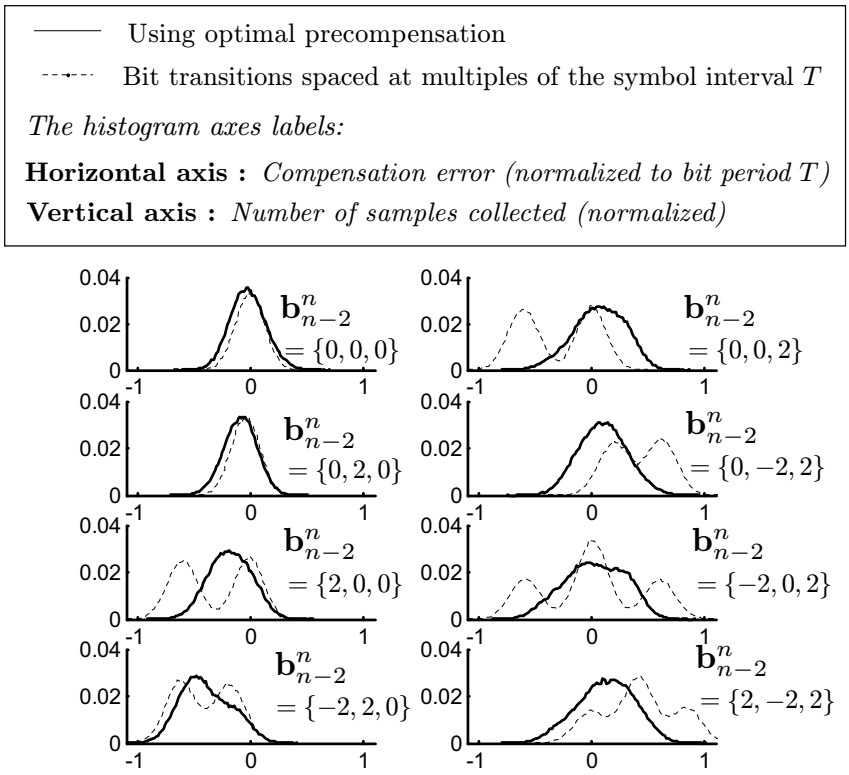


Figure 6.5: Compensation error histograms for different bit patterns $\mathbf{b}_{n-2}^n$. Two sets of compensation error histograms are shown, the first obtained when optimal precompensation values (computed using our method) was used, and the second obtained when writing bits such that the distance between any two transitions is a mulitiple of the symbol interval T. For most bit patterns $\mathbf{b}_{n-2}^n$, the first set of histograms show multiple "peaks". Use of optimal precompensation values seems to help reduce these "peaks", thus making the error more "Gaussian-like".

policies defined in the previous sections do not account for any look-ahead, since all controls are decided based on the incoming bit.

Table 6.1 compares MSE per bit obtained by various precompensation schemes. We note that our dynamic programming algorithm outperforms the simple look-ahead strategy of writing dibits further apart. However, it incurs a larger MSE per bit than if we write all bit transitions located at the ends of transition runs further apart. This motivates further investigation of dynamic programming techniques that incorporate some look-ahead; this will be a topic of future research.

Method	MSE per bit ($\times 10^{-2}$)
Alignment with sampling instances	17.10
Q-learning (Dynamic Programming)	7.38
Writing only dibits further apart	9.95
Writing bit transitions located at ends of transition runs further apart	3.95

Table 6.1: Comparison of the MSE per bit obtained for various precompensation schemes. We note that because we do not account for look-ahead decisions in our dynamic programming, we get outperformed by an intuitive look-ahead method, which is to write all bit transitions located at the ends of transition runs further apart.

6.5 Summary

In this chapter, we considered a realistic situation, in which the functional forms for the nonlinearities are unknown, and equalization techniques are used to shorten the ISI lengths. We proposed to use Q-learning techniques, which are based on the dynamic programming principle, to compute optimal precompensation. We focused specifically on the Q-learning technique for the discounted-cost problem. In order to use the Q-leaning algorithm, first we need to measure the NLTS. NLTS measurements can be done by methods proposed in the literature. We also considered the application of Q-learning techniques, when an equalizer is used to reduce the ISI length. This is done by assuming an appropriate ISI extension length. The Q-learning algorithm was tested using computer simulations. The Q-learning estimates were corroborated using Monte Carlo simulations, and were found to be relatively unbiased. The ISI length assumption was tested, and for a chosen EPR4 target, the value $\delta = 1$ gives good compromise between performance and complexity. We performed simple studies to observe the remnant noise after optimal precompensation is applied, in which we found that the noise becomes more "Gaussian-like". Finally, we performed comparisons with look-ahead policies, and found that our dynamic programming approach is outperformed by a simple precompensation scheme that incorporates look-ahead. This motivates further study to incorporate look-ahead strategies in our dynamic programming approach.

Chapter 7

Conclusion and Further Work

7.1 Current Results

There are two main results in this thesis. The first main result is the theoretical computation of optimal precompensation that minimizes the MSE between the nonlinear readback signal and a desired linear one. We proposed a dynamic programming approach, and formulated the problem to fit into such a framework. We surveyed various dynamic programming methods in Chapter 3. We showed in Chapter 3 how finite-horizon dynamic programing can be used to find the exact optimal solution, which turns out to be a non-stationary dynamic programming policy. We argued that finite-horizon dynamic programming incurs a high complexity cost. In Chapter 3, we also proposed to use infinite-horizon dynamic programming techniques, namely the discounted-cost and average-cost techniques, that deliver reduced complexity and stationary policies.

We found extraction of optimal precompensation values from the dyanmic programming policies to be complicated, and hence in Section 4.2, we proposed a suboptimal solution.

The second main result is the blind estimation method presented in Chapter 6. We proposed the use of the well known Q-learning technique to compute optimal precompensation when the channel characteristics are unknown. We pointed out that when adopting such an approach, we need to estimate the functional form of NLTS, which can be easily done using currently known NLTS measurement methods. We considered a practical application of the Q-learning technique, where equalizers are used to reduce the ISI. We pointed out the relevant assumptions required when adopting such an approach. Simulation results, obtained using the Q-learning technique to compensate for the well known partial response EPR4 target, were presented in Chapter 6.

7.2 Further Work

In this initial effort, we used the MSE as the optimality criterion to design dynamic programming techniques to solve the precompensation problem. Another optimality criterion that is of great interest would be the bit-error rate (BER). It may be possible to compute optimal precompensation, to improve the BER performance of currently known bit detectors. Perhaps, it may also be possible to compute optimal precompensation to allow new, simple detectors that outperform presently known counterparts. It would also be interesting to study how to compute optimal precompensation that maximizes the information rate of the channel. Perhaps, we would be able to increase the capacity of the channel using intelligent write precompensation.

In our initial approach to apply dynamic programming, we formulated our policies without any bit look-ahead whatsoever. In Chapter 6, we pointed out that our current approach will perform worse than optimization strategies with bit look-aheads. This motivates an extension of our methods towards dynamic

programming for policies with bit look-aheads. While we can anticipate an obvious improvement in performance, we foresee that such an approach will be non-trival. To our current knowledge, dynamic programming techniques that perform such a task do not exist in the literature.

We also observed in Chapter 6 that it typically requires a lot of data for the Q-learning algorithm to converge. This is because we need to realize all possible state transitions in the FSM, and our FSM typically has a large number of states. In magnetic recording, since the size of the typical data file is in the order of megabytes, writing a few million data bits can be done really fast. However, it would be convenient if we could reduce the required number of data bits. In the dynamic programming literature, there exist methods that only explore state transitions which are possibly optimal [5], and ignore state transitions which obviously will not be part of the optimal policy. This reduces the number of state transitions we need to realize, thus reducing the data length required by Q-learning.

Finally, our results could be strengthened by testing the derived algorithms with realistic waveforms obtained from a spin-stand. During this work, we mainly focused on algorithm development. Obviously these algorithms eventually need to be tested on spin-stand waveforms.

Appendix A

Polynomials Used to Model PE Functions

In this appendix, we give the coefficients of the polynomials used to model the PE functions shown in Sections 5.1 and 6.4.

Section 6.4 (*Figure 6.1: SNR=19 dB*)

Order	7	6	5	4	3	2
Coef	-0.27079860	2.01226812	-5.85592704	8.52781928	-6.71345891	2.73208312

Order	1	0
Coef	0.17445192	0.25267821

Section 5.1 (*Figure 5.2*), and Section 6.4 (*Figure 6.1: SNR=24 dB*)

Order	7	6	5	4	3	2
Coef	-0.08345976	0.86106975	-3.18699273	5.61751907	-5.30899702	2.79987281

Order	1	0
Coef	-0.06050535	0.07290768

Section 6.4: (*Figure 6.1: SNR=30 dB*)

Order	7	6	5	4	3	2
Coef	0.78167266	-5.46801959	15.11652385	-20.90184342	14.56579915	-3.88242199

Order	1	0
Coef	0.30244275	-0.00040084

Appendix B

Linear Equalizer Tap Coefficients

In this appendix, we give the tap coefficients of the 31-tap linear equalizer, used to obtain the simulation results shown in Section 6.4.

Taps	Values
1 & 31	-0.004003508
2 & 30	-0.001530436
3 & 29	-0.004363056
4 & 28	-0.004411582
5 & 27	-0.006182235
6 & 26	-0.007460240
7 & 25	-0.009645008
8 & 24	-0.013447671
9 & 23	-0.011296905
10 & 22	-0.043294927
11 & 21	0.042990970
12 & 20	-0.295731673
13 & 19	0.669950305
14 & 18	-1.919653601
15 & 17	-0.232183893
16	5.471995798

Bibliography

[1] T. C. Arnoldussen, "Thin-film recording media," *Proceedings of the IEEE*, vol. 74, no. 11, pp. 1526–1539, Nov. 1986.

[2] T. C. Arnoldussen and J. G. Zhu, "Nonlinear behavior of magnetoresistive heads," *IEEE Trans. Magn.*, vol. 34, no. 1, pp. 36–39, Jan. 1998.

[3] J. W. M. Bergmans, *Digital Baseband Transmission and Recording*, Kluwer Academic Publishers, 1st ed., 1996, chap. 2,6,7.

[4] H. N. Bertram, *Theory of Magnetic Recording*, Cambridge University Press, 2nd ed., 1994, chap. 1,9.

[5] H. N. Bertram and M. Williams, "SNR and density limit estimates: A comparison of longitudinal and perpendicular recording," *IEEE Trans. Magn.*, vol. 36, no. 1, pp. 4–9, Jan. 2000.

[6] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume I*, Athena Scientific, 2nd ed., 2000, chap. 1.

[7] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume II*, Athena Scientific, 2nd ed., 2000, chap. 1,4.

[8] M. Blaum, "An introduction to error-correcting codes," in *Coding and Signal Processing Techniques for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas, Eds. CRC Press, 1st ed., 2005, chap. 9.

[9] H. Cai, "Magnetoresistive read nonlinearity correction by a frequency-domain approach," *IEEE Trans. Magn.*, vol. 35, no. 6, pp. 4532–4534, Nov. 1999.

[10] H. Cai, "New frequency-domain technique for joint measurement of nonlinear transition shift and partial erasure," *IEEE Trans. Magn.*, vol. 35, no. 6, pp. 4535–4537, Nov. 1999.

[11] J. Caroselli, J. Fitzpatrick, and J. Wolf, "A simple model for transition interactions with the microtrack model," in *Proc. IEEE Intl. Conf. Commun. (ICC)*, Atlanta, GA, Jun. 1998, vol. 2, pp. 942–946.

[12] X. Che, "Nonlinearity measurements and write precompensation studies for a PRML recording channel," *IEEE Trans. Magn.*, vol. 31, no. 6, pp. 3021–3026, Nov. 1995.

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

[14] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *Journal of Machine Learning Research*, vol. 5, pp. 1–25, Dec. 2003.

[15] P. C. I. Fang, X. Feng, T. T. L. Lam, and Z. H. Lin, "Process for measuring nonlinear transition shift NLTS at high recording densities with a giant mag-

netoresistive GMR head," *US Patent Application*, no. US 2003/0179478 A1, Sep. 2003.

[16] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the prescence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 18, no. 3, pp. 363–378, May 1972.

[17] R. Hermann, "Volterra modeling of digital magnetic saturation recording channels," *IEEE Trans. Magn.*, vol. 26, no. 5, pp. 2125–2127, Sep. 1990.

[18] T. D. Howell, D. P. McCown, T. A. Diola, Y. S. Tang, K. R. Hense, and R. L. Gee, "Error rate performance of experimental gigabit per square inch recording components," *IEEE Trans. Magn.*, vol. 29, no. 5, pp. 2298–2302, Sep. 1990.

[19] P. Kabal and S. Pasupathy, "Partial response signaling," *IEEE Trans. Commun.*, vol. 23, no. 9, pp. 921–934, Sep. 1975.

[20] A. Kavcic, "Soft-output detector for channels with intersymbol interference and Markov noise memory," in *Proc. IEEE Intl. Conf. Global Telecommun. (GLOBECOM)*, 1999, Rio de Jeneiro, Dec. 1999, vol. 1b, pp. 728–732.

[21] A. Kavcic and J. M. F. Moura, "Statistical study of zig-zag transition boundaries in longitudinal digital magnetic recording," *IEEE Trans. Magn.*, vol. 33, no. 6, pp. 4482–4491, Nov. 1997.

[22] A. Kavcic and J. M. F. Moura, "The Viterbi algorithm and Markov noise memory," *IEEE Trans. Inform. Theory*, vol. 46, no. 5, pp. 291–301, Jan. 2000.

[23] A. Kavcic and A. Patapoutian, "A signal-dependent autoregressive channel model," *IEEE Trans. Magn.*, vol. 35, no. 5, pp. 2316–2318, Sep. 1999.

[24] I. Lee, T. Yamauchi, and J. M. Cioffi, "Performance comparison of receivers in a simple partial erasure model," *IEEE Trans. Magn.*, vol. 30, no. 4, pp. 1465–1469, Jul. 1994.

[25] J. Lee and J. Lee, "A simplified noise-predictive partial response maximum likelihood detection using M-algorithm for perpendicular magnetic recording channels," *IEEE Trans. Magn.*, vol. 41, no. 2, pp. 1064 – 1066, Feb. 2005.

[26] G. H. Lin and H. N. Bertram, "Experimental studies of noise autocorrelation in thin film media," *IEEE Trans. Magn.*, vol. 29, no. 6, pp. 3697–3699, Nov. 1993.

[27] J. Moon, "Discrete-time modeling of transition-noise-dominant channels and study of detection performance," *IEEE Trans. Magn.*, vol. 27, no. 6, pp. 4573–4578, Nov. 1991.

[28] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Trans. Magn.*, vol. 32, no. 5, pp. 3992–3994, Sep. 1996.

[29] Y. Okamoto, H. Sumiyoshi, T. Kishigami, M. Akamatsu, H. Osawa, H. Saito, H. Muraoka, and Y. Nakamura, "A study of PRML systems for perpendicular

recording using double layered medium," *IEEE Trans. Magn.*, vol. 36, no.5, pp. 2164–2166, Sep. 2000.

[30] D. Palmer, "A brief history of magnetic storage," in *Coding and Signal Processing Techniques for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas Eds. CRC Press, 1st ed., 2005, chap. 1.

[31] D. Palmer, J. Hong, D. Stanek, and R. Wood, "Characterization of the read/write process for magnetic recording," *IEEE Trans. Magn.*, vol. 31, no. 2, pp. 1071–1076, Mar. 1995.

[32] D. Palmer, P. Ziperovich, R. Wood, and T. Howell, "Identification of nonlinear write effects using psuedorandom sequences," *IEEE Trans. Magn.*, vol. 23, no. 5, pp. 2377–2379, Sep. 1987.

[33] W. H. Press, S. A. Teukolsky, W. Vetterling, and B. P. Flannery, *Numerical recipies in C*, Cambridge University Press, 2nd ed., 1996, chap. 10.

[34] K. Senanan and R. H. Victora, "Theoretical study of nonlinear transition shift in double-layer perpendicular media," *IEEE Trans. Magn.*, vol. 38, pp. 1664–1669, Jul. 2002.

[35] Y. Tang and C. Tsang, "A technique for measuring non-linear bit shift," *IEEE Trans. Magn.*, vol. 27, no. 6, pp. 5316–5318, Nov. 1991.

[36] A. Taratorin, *Characterization of magnetic recording systems*, Guzik technical publications, 1st ed., 1996, chap. 5-7.

[37] A. Taratorin, D. Cheng, P. Arnett, R. Olson, J. Diola, T. amd Fitzpatrick, S. Wang, and B. Wilson, "Intra- and inter-pattern non-linearities in high density magnetic recording," *IEEE Trans. Magn.*, vol. 34, no. 1, pp. 45–50, Jan. 1998.

[38] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*, vol. 16, no. 3, pp. 185–202, Sep. 1994.

[39] C. J. C. H. Watkins and D. Peter, "Q-learning," *Machine Learning*, vol. 8, pp. 279–292, May 1992.

[40] A. J. Wijngaarden and E. Soljanin, "A combinatorial technique for constructing high-rate MTR-RLL codes," *IEEE J. Select. Areas Commun.*, vol. 19, no. 4, pp. 582–588, Apr. 2001.

[41] W. Zeng and J. Moon, "Modified Viterbi algorithm for a jitter-dominant $1 - D^2$ channel," *IEEE Trans. Magn.*, vol. 28, no. 5, pp. 2895–2897, Sep. 1992.

[42] H. Zhou, T. Roscamp, R. Gustafon, E. Boerner, and R. Chantrell, "Physics of longitudinal and perpendicular recording," in *Coding and Signal Processing Techniques for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas, Eds. CRC Press, 1st ed., 2005, chap. 2.

[43] W. Zhu, D. Kaiser, J. Judy, and D. Palmer, "Experimental study of reader nonlinearity in perpendicular recording using pseudorandom sequences," *IEEE Trans. Magn.*, vol. 39, no. 5, pp. 2636–2638, Sep. 2003.

List of Publications

[1] F. Lim and A. Kavcic, "Optimal pre-compensation for partial erasure and non-linear transition shift in magnetic recording using dynamic programming," in *Proc. IEEE. Intl. Conf. Global Telecommun. (GLOBECOM)*, 2005, St Louis, MO, Nov. 2005.

[2] F. Lim and A. Kavcic, "Optimal precompensation for nonlinearities in longitudinal magnetic recording using dynamic programming," to appear in *Intl. J. Prod. Develop*, 2006.

Bibliography

[1] T. C. Arnoldussen, "Thin-film recording media," *Proceedings of the IEEE*, vol. 74, no. 11, pp. 1526–1539, Nov. 1986.

[2] T. C. Arnoldussen and J. G. Zhu, "Nonlinear behavior of magnetoresistive heads," *IEEE Trans. on Magn.*, vol. 34, no. 1, pp. 36–39, Jan. 1998.

[3] J. W. M. Bergmans, *Digital Baseband Transmission and Recording*, Kluwer Academic Publishers, 1996.

[4] H. N. Bertram, *Theory of Magnetic Recording*, Cambridge University Press, 2nd edn., 1994.

[5] D. P. Bertsekas, *Dynamic Programming and Optimal Control, Volume II*, Athena Scientific, chap. 1.

[6] H. Cai, "Magnetoresistive read nonlinearity correction by a frequency-domain approach," *IEEE Trans. on Magn.*, vol. 35, no. 6, pp. 4532–4534, Nov. 1999.

[7] H. Cai, "New frequency-domain technique for joint measurement of nonlinear transition shift and partial erasure," *IEEE Trans. on Magn.*, vol. 35, no. 6, pp. 4535–4537, Nov. 1999.

[8] J. Caroselli and J. Wolf, "Applications of a new simulation model for media noise limited magnetic recording channels," *IEEE Trans. on Magn.*, vol. 32, no. 5, pp. 3917–3919, Sep. 1996.

[9] X. Che, "Nonlinearity measurements and write precompensation studies for a PRML recording channel," *IEEE Trans. on Magn.*, vol. 31, no. 6, pp. 3021–3026, Nov. 1995.

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

[11] R. W. D. Palmer, P. Ziperovich and T. Howell, "Identification of nonlinear write effects using psuedorandom sequences," *IEEE Trans. on Magn.*, vol. 23, no. 5, pp. 2377–2379, Sep. 1987.

[12] E. Even-Dar and Y. Mansour, "Learning rates for Q-learning," *Journal of Machine Learning Research*, vol. 5, pp. 1–25, Dec. 2003.

[13] G. D. Forney, "Maximum-likelihood sequence estimation of digital sequences in the prescence of intersymbol interference," *IEEE Trans. on Inform. Theory*, vol. 18, no. 3, pp. 363–378, May 1972.

[14] R. Hermann, "Volterra modeling of digital magnetic saturation recording channels," *IEEE Trans. on Magn.*, vol. 26, no. 5, pp. 2125–2127, Sep. 1990.

[15] T. D. Howell, D. P. McCown, T. A. Diola, Y. S. Tang, K. R. Hense, and R. L. Gee, "Error rate performance of experimental gigabit per square inch

recording components," *IEEE Trans. on Magn.*, vol. 29, no. 5, pp. 2298–2302, 1990.

[16] P. Kabal and S. Pasupathy, "Partial response signaling," *IEEE Trans. on Comm.*, vol. 88, no. 8, pp. 921–934, Sep. 1975.

[17] A. Kavcic, "Soft-output detector for channels with intersymbol interference and markov noise memory," *IEEE Trans. on Magn.*, vol. 39, no. 5, pp. 2636–2638, Sep. 2003.

[18] A. Kavcic and J. Moura, "Statistical study of zig-zag transition boundaries in longitudinal digital magnetic recording," *IEEE Trans. on Magn.*, vol. 33, no. 6, pp. 4482–4491, Nov. 1997.

[19] A. Kavcic and J. Moura, "The Viterbi algorithm and Markov noise memory," *IEEE Trans. on Inform. Theory*, vol. 46, no. 5, pp. 291–301, Jan. 2000.

[20] A. Kavcic and A. Patapoutian, "A signal-dependent autoregressive channel model," *IEEE Trans. on Magn.*, vol. 35, no. 5, pp. 2316–2318, Sep. 1999.

[21] G. H. Lin and H. N. Bertram, "Experimental studies of noise autocorrelation in thin film media," *IEEE Trans. on Magn.*, vol. 29, no. 6, pp. 3697–3699, Nov. 1993.

[22] J. Moon, "Discrete-time modeling of transition-noise-dominant channels and study of detection performance," *IEEE Trans. on Magn.*, vol. 27, no. 6, pp. 4573–4578, Nov. 1991.

[23] J. Moon and B. Brickner, "Maximum transition run codes for data storage systems," *IEEE Trans. on Magn.*, vol. 32, no. 5, pp. 3992–3994, Sep. 1996.

[24] D. Palmer, J. Hong, D. Stanek, and R. Wood, "Characterization of the read/write process for magnetic recording," *IEEE Trans. on Magn.*, vol. 31, no. 2, pp. 1071–1076, Mar. 1995.

[25] W. H. Press, S. A. Teukolsky, and W. Vetterling, *Numerical recipies in C*, Cambridge University Press, 1996.

[26] Y. Tang and C. Tsang, "A technique for measuring non-linear bit shift," *IEEE Trans. on Magn.*, vol. 27, no. 6, pp. 5316–5318, Nov. 1991.

[27] A. Taratorin, *Characterization of magnetic recording systems*, Guzik technical publications, 1st edn., 1996.

[28] A. Taratorin, D. Cheng, P. Arnett, R. Olson, J. Diola, T. amd Fitzpatrick, S. Wang, and B. Wilson, "Intra- and inter-pattern non-linearities in high density magnetic recording," *IEEE Trans. on Magn.*, vol. 34, no. 1, pp. 45–50, Jan. 1998.

[29] J. N. Tsitsiklis, "Asynchronous stochastic approximation and Q-learning," *Machine Learning*, vol. 16, pp. 185–202, 1994.

[30] C. J. C. H. Watkins, "Q-learning," *Machine Learning*, vol. 8, no. 6, pp. 279–292, 1992.

[31] W. Zeng and J. Moon, "Modified Viterbi algorithm for a jitter-dominant $1-d^2$ channel," *IEEE Trans. on Magn.*, vol. 28, no. 5, pp. 2895–2897, Sep. 1992.

[32] W. Zhu, D. Kaiser, J. Judy, and D. Palmer, "Experimental study of reader nonlinearity in perpendicular recording using pseudorandom sequences," *IEEE Trans. on Magn.*, vol. 39, no. 5, pp. 2636–2638, Sep. 2003.