DESIGN OF EFFICIENT CONSTRAINED CODES AND PARITY-CHECK CODES FOR PERPENDICULAR MAGNETIC RECORDING CHANNELS

MOULAY RACHID ELIDRISSI

NATIONAL UNIVERSITY OF SINGAPORE

2004

DESIGN OF EFFICIENT CONSTRAINED CODES AND PARITY-CHECK CODES FOR PERPENDICULAR MAGNETIC RECORDING CHANNELS

MOULAY RACHID ELIDRISSI

(B. Sc. (Hons.), TELECOM INT)

A THESIS SUBMITTED

FOR THE DEGREE OF MASTER OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER

ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2004

Acknowledgments

I would like to express my greatest gratitude to Dr. George Mathew for all his invaluable guidance, support and patience throughout the work of this thesis. I am sincerely thankful to him for conscientiously teaching me, guiding me, and reviewing my drafts. His diligence and enthusiasm towards research work has inspired me greatly. I have gained tremendous wisdom through the discussions we have had, and I am deeply grateful to him. He is truly a brilliant man, and it has been an honor and privilege working under his supervision.

I would like to express my gratitude to my friend Mr Bruno Low whose support and encouragement have always accompanied me along the way.

I am also grateful to my family which, from France, has constantly provided me the support I needed to complete my master's degree in Singapore.

Last but not least, I would like to thank all the staff in Data Storage Institute, who helped me in one way or another.

Contents

Summary

Because of the potential of perpendicular magnetic recording to achieve much higher recording densities than longitudinal recording, the recent years have witnessed a surge in research activities in perpendicular recording. Whereas most of the techniques developed for longitudinal recording are also applicable to perpendicular recording, they need to be reinvestigated to ensure optimum performance for perpendicular recording. The combination of a partial response (PR) equalizer followed by the Viterbi algorithm based maximum likelihood (ML) type sequence detection (PRML) is currently the most commonly used signal detection technique at high recording densities. To further increase the performance, distance-enhancing modulation codes are used. These codes, at the cost of efficiency, help to gain performance by eliminating the data patterns that support dominant error mechanisms in the detector. To minimize the loss in efficiency, especially at high densities, an approach that is widely being adopted is to use weaker distance-enhancing codes in combination with parity-check error correcting codes. The approach of parity-based post-processing to detect and correct errors is now widely being adopted since it offers a good compromise between performance and complexity. Therefore, the research work undertaken in this thesis is aimed at developing efficient constrained parity-check codes and optimum post-processing approaches for perpendicular recording.

Analysis of the error events at the Viterbi detector (VD) output shows that most of the dominant error events have a certain common structure. This motivates the design of a suitable and efficient distance-enhancing code. This code is then combined with a parity-check code. Moreover, the special structure of the parity-check code results in a post-processor that is computationally much simpler than conventional post-processors. Simulated for a perfectly equalized PR channel, the novel constrained parity-check code provides significant improvement in bit error rate (BER) performance.

Parity-based post-processors are based on the principle of optimum receiver for multiple signals detection in communication theory. Because most post-processors in current systems are based on ML criterion, it is proposed in this thesis to investigate the performance of post-processors based on maximum *a posteriori* (MAP) criterion. We have derived analytical expressions for the comparison of MAP and ML post-processors. However, because the novel constrained parity-check code results in error events that have comparable prior probabilities, the performance of MAP and ML post-processors for this code turn out to be similar.

Weakly constrained modulation codes in combination with parity-check codes have received much interest because of the good trade-off they offer between coding gain and rate loss. Runlength constraints characterized with a list of forbidden data strings, known as forbidden list (FL) constraints, represent an effective way of generating high-rate constrained codes. Because of the flexibility offered by weak FL constraints, it is proposed in this thesis to search for the FL constraints that simultaneously lead to acceptable BER performance and separate the prior probabilities for the MAP post-processor. After identifying eligible FL constraints, we generate the constrained data with a suitably designed maxentropic Markov source. Simulations on real channels show attractive performance gain with this new code.

Nomenclature

$\underline{\varepsilon}$	error string associated with an error event $\underline{e}$
$\eta(n)$	noise component of the signal at the output of the equalizer
$\mathcal{L}$	list of forbidden strings $\underline{s}_j$
$\pi(e)$	probability of the data patterns which support the error event $\underline{e}$
$\upsilon(m)$	channel noise
$c(n)$	input data bits in NRZ{-1,1} format
$\hat{c}(n)$	channel bits detected by Viterbi detector
D	one channel bit delay operator
D_u	user density
d_e	Euclidean distance associated with the error event $\underline{e}$
$\underline{e}$	error event
g_k	taps of the partial-response target
$h(t),\ h_i$	bit response of the recording channel
$h_s(t)$	step response of the recording channel
L	oversampling factor
N_g	length of the partial response target
P_b	bit error rate
R	code rate

$\underline{s}_j$	forbidden string which constrain the channel data bits $c(n)$
T	duration of one channel bit
$w(e)$	Hamming weight of the error event $\underline{e}$
w_i	equalizer taps
$x(n)$	output of the equalized recording channel
AWGN	additive white Gaussian noise
BER	bit error rate
FL	forbidden list
GPR	generalized partial response
ISI	intersymbol interference
MAP	maximum *a posterior*
ML	maximum likelihood
MLSD	maximum likelihood sequence detection
MTR	maximum transition run
NRZI	non-return-to-zero inverse
NRZ	non-return-to-zero
PC	parity-check
PPP	parity-based post-processor
PR	partial response
PRML	partial response maximum likelihood
RLL	runlength limited
SNR	signal-to-noise ratio
VD	Viterbi detector

List of Figures

List of Tables

Chapter 1

Introduction

In this chapter, we present a very brief overview of magnetic data storage systems from a signal processing perspective. After introducing perpendicular magnetic recording and the different signal detection techniques, we focus on constrained codes and parity-check codes which are used for improving the detection performance in recording systems. This review is aimed at motivating the research work reported in this thesis. The chapter concludes with a summary of the main contributions and the organization of the thesis.

1.1 Magnetic Data Storage

The advent of the information age has triggered a tremendous demand for mass data storage. Demand for storage capacity is doubling every 18 months, which accounts for the importance of data storage as the central component of information technology infrastructure. There is great economic as well as technological interest in data storage. The current data storage industry is comprised of three technologies: magnetic storage, optical storage, and semiconductor memory. Storage media for magnetic recording can be hard disks, tapes, floppy disks and cards. Currently, magnetic storage has a clear leading edge over the other two technologies in providing high storage capacity devices at low costs and high flexibility. To maintain this edge and meet the increasing demand for data storage capacity posed by the information technology era, the currently used longitudinal magnetic recording technology has shown phenomenal growth rate in storage capacity in the last two decades. Even though longitudinal technology has well exceeded 100 Gbits/in^2, it is well understood that the growth rate cannot be maintained further due to the so-called ‘super paramagnetic effect’ which arises from the inability of the magnetic medium to hold the magnetization pattern at very high densities [1]. As a result, there has been intense research in perpendicular recording because of its potential to achieve much higher recording densities than longitudinal recording by pushing the threshold up for the occurrence of super paramagnetic effect [2]. The perpendicular recording has already achieved about 150 Gbits/in^2. Although the explosive growth in recording density has been mainly due to the technological improvements made in the design of head-media system, sophisticated coding and signal processing techniques are very essential to support and enhance high density recording [50]. The scope of this thesis is limited to the coding and signal processing aspects of perpendicular magnetic recording systems. In particular, the objective is to develop efficient constrained parity-check codes for improving the detection performance.

1.2 Perpendicular Magnetic Recording

Figure 1.1 shows the block schematic of a digital magnetic recording system. The ECC (error control coding) encoder incorporates error detection and error correction capabilities into the input data [25,39]. The purpose of modulation encoder is to match the data to the recording channel characteristics, improve detection performance and ensure proper working of the control loops (*e.g.* timing/gain recovery) at the receiver [26,53,51]. The write circuits convert the coded data sequence into a write-current which in turn drives the write-head, to magnetize the storage medium to record the coded data. The front-end circuits condition the replay signal generated by the read head, *i.e.* limit the noise bandwidth, regulate dynamic range, compensate for nonlinearities etc [41]. The equalizer shapes the signal according to certain criteria so that the detector is able to recover the stored data with as few errors as possible [13,27]. The data detector recovers the stored encoded data and passes it to the modulation and ECC decoders for recovering the original user data. Not shown explicitly in Figure 1.1 are the control loops required for doing timing recovery [44], gain control [41], DC offset cancellation and adaptive equalization [27].

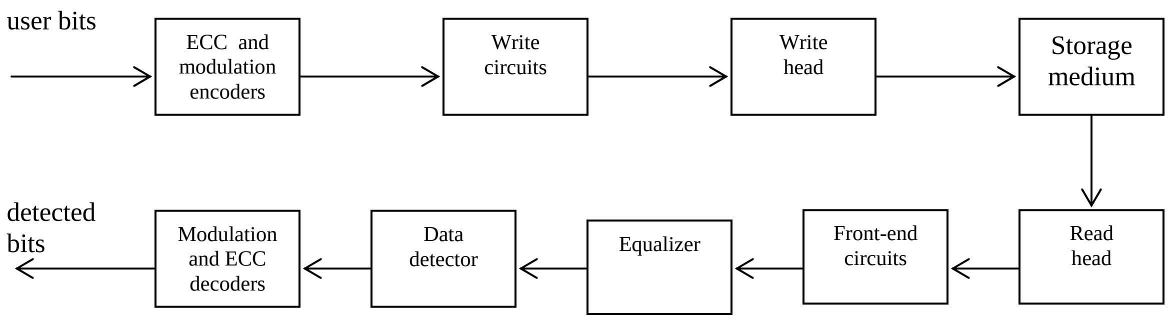

Figure 1.1: Block schematic of a digital magnetic recording system.

The main features expected from a recording system are high storage density, high data rate, good reliability and low power requirement. However, the noise power increases with data rate. Further, as density increases, the signal-to-noise ratio (SNR) of the read-back signal decreases and channel distortions increase. These effects reduce significantly the reliability and the achievable recording density of the system [42].

Several strategies have been developed to improve recording density in magnetic recording systems. Because of the potential for supporting high density recording, there has been intense research in perpendicular recording. As shown in Figure 1.2, the magnetization direction on the storage medium is perpendicular to the plane of the medium in perpendicular recording systems. It is expected that the perpendicular recording technology will replace the longitudinal technology in the coming few years. Wood [3] shows that double-layer perpendicular magnetic recording systems are capable of achieving recording densities as high as 1 Tbits/in^2. While the increase in density has been mainly due to the technological improvements in the recording medium and read/write heads, the coding and signal processing techniques have become very crucial to ensure reliable data recovery despite the serious degradation in channel quality (*e.g.* noise, distortions, SNR) under aggressive recording conditions [42].

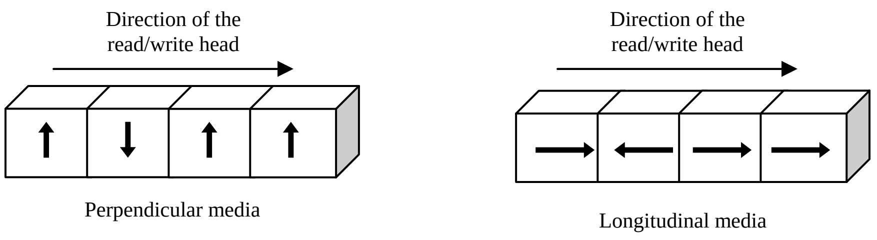
Direction of the read/write head
Perpendicular media
Direction of the read/write head
Longitudinal media

Figure 1.2: Magnetization pattern of the storage medium for perpendicular and longitudinal magnetic recording.

Regardless of the direction of the medium magnetization (see Figure 1.2), recording systems need to tackle similar set of issues, *i.e.* intersymbol interference (ISI), media noise, head and electronics noise, and various nonlinear distortions. Therefore, the coding and signal processing techniques developed for longitudinal recording systems should also be suitable for perpendicular recording systems. However, detailed characteristics, such as the spectrum of the channel response and the nature of nonlinearities, of longitudinal and perpendicular systems differ greatly [43]. Unlike in longitudinal recording systems, the read-back signal has a DC component, and this affects the characteristic of the detector. In view of these differences, it becomes necessary to re-investigate the coding and signal processing techniques so as to achieve the best performance from perpendicular recording systems.

In view of the scope of the research work undertaken in this thesis, the brief reviews given in Sections 1.3 and 1.4 are restricted to equalization, detection and coding techniques used in magnetic recording system.

1.3 Detection Techniques for Magnetic Recording

There are a number of techniques developed for data detection on magnetic recording channels. The most widely used detection method in commercial products till late 1980’s was the peak detection method [48]. At low recording densities, in longitudinal recording, transitions in the polarity of the input data sequence result in reasonably distinct peaks in the read-back signal. Therefore, the principle of peak detector is to detect the presence of transitions in the input data by detecting the peaks in the read-back signal. The peaks are located by differentiating the read-back signal and passing it through a ‘zero-crossing detector’. To minimize detection errors, a threshold test is also done on the signal to eliminate spurious zero-crossings caused by noise. Clearly, this detector is very simple to implement.

The peak detector works best at low recording densities. However, as the density increases, the peak detector is not suitable anymore. In 1990, IBM developed a new detection technique, known as PRML, based on the principles of partial response (PR) signaling and maximum likelihood (ML) detection [13]. PRML detectors, depicted in Figure 1.3, are able to support much higher recording densities than peak detectors and they have become the most widely used detection technique in current commercial hard-disk drives. PRML detectors include a PR equalizer to shape the recording channel to a suitably chosen target response [45], called the PR target, and a maximum likelihood sequence detector (MLSD) [11] to recover the recorded bit sequence. PR equalization serves to reduce the complexity of the MLSD which is implemented using the Viterbi algorithm [46]. The acronym 'PRML' may lead to confusion. Indeed, the Viterbi detector is not the optimal detector for this system since the noise at its input is colored due to the equalizer. However, the industry as well as the recording literature have been using this acronym despite its inaccuracy [5,13]. Therefore, for the sake of tradition, we will also use this acronym in this thesis and remember that the underlying Viterbi detector is not optimal.

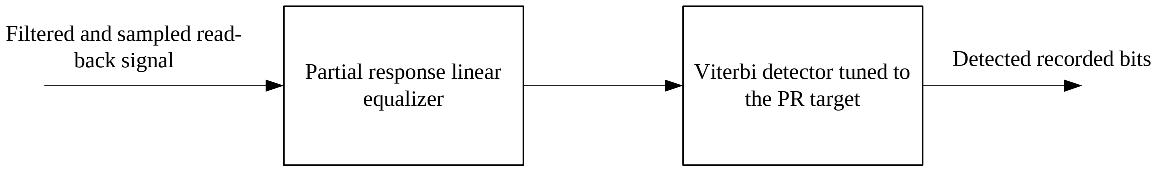
Filtered and sampled read-back signal
Partial response linear equalizer
Viterbi detector tuned to the PR target
Detected recorded bits

Figure 1.3: Partial-response maximum likelihood (PRML) detection technique.

Noise enhancement and noise correlation degrade the performance of PRML schemes [5]. The performance of PRML detectors can be improved if the noise at the input of the Viterbi detector is whitened. Chevillat *et al.* [4] and Coker *et al.* [5] have showed that the performance of the Viterbi detector can be enhanced by attaching a noise-whitening filter at the output of the PR equalizer. The combination of the PR equalizer, the noise whitening filter and the modified Viterbi detector, shown in Figure 1.4, is called noise-predictive maximum-likelihood (NPML) detector [5].

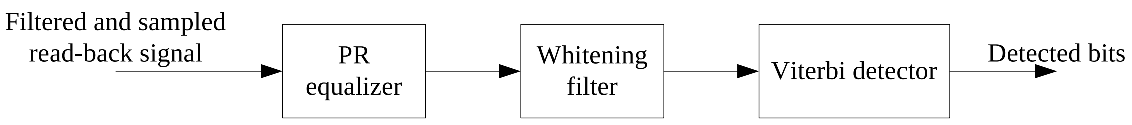

Figure 1.4: Block diagram of noise-predictive maximum-likelihood (NPML) detector.

The use of whitening filter results in increasing the length (*i.e.* the number of coefficients) of the effective PR target. Since the complexity of the Viterbi detector grows exponentially with the PR target length, it becomes necessary to use some complexity reduction approaches while implementing the NPML. The approach used in NPML is to reduce the number of states in the Viterbi trellis by using bits from the survivor paths in the trellis [5,47].

The issue of noise enhancement can also be dealt with a detector based on the principle of decision feedback equalization (DFE) [49]. DFE detection schemes, shown in Figure 1.5, were developed in parallel with PRML schemes. The joint action of the forward and feedback equalizers results in complete cancellation of intersymbol interference (ISI) at the slicer input, while not causing noise enhancement. On the other hand, complete cancellation of ISI results in loss in achievable performance compared to the PRML approach. Another disadvantage of DFE is the phenomenon of error propagation, which arises because the feedback equalizer uses the already detected bits to cancel part of the ISI. Error propagation is caused by incorrect past decisions. As compared to PRML schemes, DFE detectors are simple in structure, have low implementation cost and processing requirement, and require no decision delay.

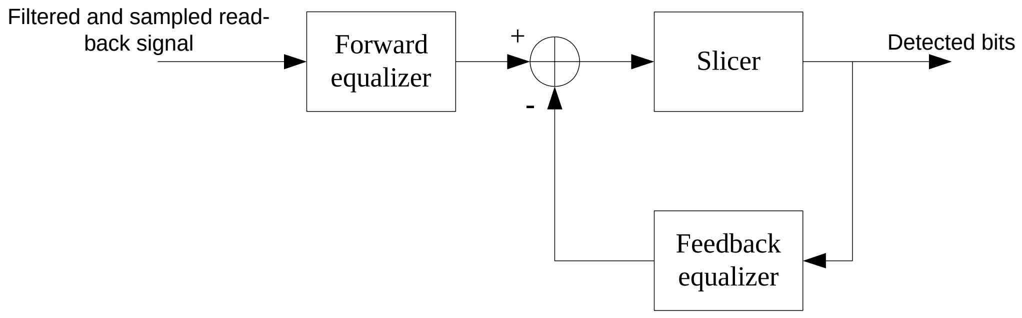

Figure 1.5: Block diagram of a detector based on feedback equalization (DFE).

1.4 Constrained Codes

In recording systems, ‘constrained codes’ is another name for ‘modulation codes’. As mentioned in Section 1.2, constrained codes play a key role in ensuring reliable data recovery. Further, a close examination of the detectors described in Section 1.3 will show the need for code constraints to be imposed on the input data. Before explaining these constraints in detail, let us give the conventions that are usually used to map an input binary sequence to the magnetization pattern along the track [51]. In non-return to zero (NRZ) convention, one direction of magnetization corresponds to a ‘1’ and the other direction corresponds to a '0'. In non-return to zero inverse (NRZI) convention, a change in the direction of magnetization corresponds to a recorded '1', whereas no change corresponds to a recorded '0'. Usually, the input sequence is first encoded using NRZI format and then transformed to the NRZ format before being fed to the write circuits. This transformation is known as 'NRZI-NRZ precoding' and its transfer function is given by $1/(1\oplus D)$ where '$\oplus$' is the Boolean XOR operator and 'D' is the 1-bit delay operator. Figure 1.6 illustrates a write-current waveform and the associated binary input data in NRZ and NRZI formats.

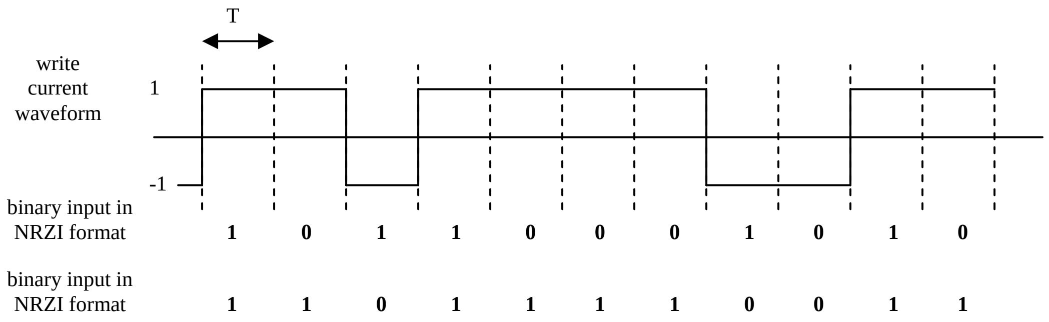

Figure 1.6: Input binary data in NRZI and NRZ formats corresponding to the given write-current waveform.

As the recording density increases, the linear as well as nonlinear ISI in the recording channel tend to increase. An immediate consequence is the degradation of the performance of simple threshold detectors such as the peak detector due to the shift in peaks and reduction in noise margin. In addition, the linearity assumption on the channel tends to fail. A class of modulation codes, called runlength-limited (RLL) codes, is particularly useful for tackling these issues [6,51]. In order to reduce the ISI (linear/nonlinear), the input data sequence in NRZI format should have a certain minimum number d of consecutive '0's between two consecutive '1's. This constraint is known as the 'd constraint'. Similarly, in order to prevent the loss of clock synchronization, the timing and gain control loops at the receiver should be updated frequently enough. This is enabled by limiting the number of consecutive '0's between two consecutive '1's to a given maximum k. This is known as the 'k constraint'. The RLL constraints are also called (d,k) constraints.

In PRML detectors, the k-constraint has the additional role of reducing the path memory requirement as well as avoiding certain catastrophic error events [13,46]. In fact, the runlength constraints in PRML detectors are specified by a more general form $(0,G/I)$ [13,51]. Here, '0' refers to the $d=0$ constraint and G refers to the k-constraint. The I-parameter specifies an additional constraint on the maximum runlength of zeros in the odd and even interleaved sequences. The I-constraint has proven to eliminate the troublesome channel input sequences that would otherwise degrade the performance of the Viterbi detector [13].

The benefits provided by the above described code constraints come at a cost that is specified by a parameter called the 'code rate'. The rate of a code is defined as $R=p/q$, $0<R<1$, specifying that groups of p bits at the encoder input are coded into groups of q bits at its output. Clearly, the code rate decreases with increase in d or decrease in k,G or I. Two main disadvantages of coding are decrease in SNR and increase in channel data rate, with decrease in code rate [12]. Therefore, it becomes very important to design codes with the maximum code rate possible, while satisfying the required code constraints.

Yet another class of constrained codes is called 'distance-enhancing codes' [7]. These codes, in addition to imposing the necessary runlength constraints, also impose special constraints for the sole purpose of enhancing the detection performance of Viterbi detector. These constraints may be formulated and implemented using time-domain or frequency-domain approaches. An example of the time-domain approach is the maximum transition run (MTR) code proposed by Brickner and Moon [8]. This code removes the dominant error patterns by eliminating input data patterns that support three or more consecutive transitions. Whereas a $(d = 1, k)$ RLL constrained code can also eliminate these data patterns, the advantage of the MTR code is that it can accomplish this with a high code rate of 4/5 compared to the $(d = 1, k)$ code whose code rate is only about 2/3. An example of the frequency-domain approach is the class of matched spectral null (MSN) codes which improve the detection performance by matching the spectra of data and channel especially at the channel nulls [26]. This has the effect of eliminating the dominant error patterns.

Because of the performance gain achieved with high-rate modulation codes, there has been intense research for designing such codes. Fitzpatrick and Modlin [9] designed high-rate distance-enhancing codes that are based on time-varying MTR constraints. Cideciyan *et al.* [52] have presented the design of high rate MTR codes for generalized PR channels. Karabed *et al.* [54] have introduced high-rate distance-enhancing codes which are defined by a list of forbidden data strings.

1.5 Parity-Check Codes and Post-Processing Techniques

In the previous section, we saw that the distance-enhancing codes (*e.g.* MTR) help to gain performance by eliminating the data patterns that support the dominant error mechanisms in the detector. The price paid to achieve this gain is the coding efficiency.

In our efforts to attain very high recording densities, it is of utmost importance to make the coding efficiency (*e.g.* code rate) as high as possible. For example, a code rate of even 8/9 is considered low according to current trends [53]. In order to minimize the loss in efficiency (in code rate), especially at high densities, an approach that is widely being adopted is to use weaker constrained codes in combination with parity-check codes [22,37,40]. Weak constrained codes do not completely prohibit the dominant error mechanisms in the detector. Instead, they restrict the number of such mechanisms and reduce their probabilities. A parity-based post-processor unit helps to detect and correct the dominant errors that remain at the detector output. Compared to the approach of joint data detection and parity decoding using the Viterbi detector, the post-processing approach of error detection and correction is very cost-effective, from the point of complexity, while not sacrificing performance. As compared to powerful ECC schemes such as turbo-codes, the parity-based post-processors represent a practically attractive trade-off between implementation complexity and performance gain. When used separately, constrained codes and parity-check codes require two distinct encoders. In order to improve the overall efficiency, it has been proposed to combine both encoders [31,34,38]. The design of the parity-check code is based on the analysis of the dominant error mechanisms in the detector. Because of their powerful error correction and error detection capability, systematic polynomial block codes have received particular interest [36,38,40]. But, these codes cannot be combined with constrained codes. Therefore, for the sake of efficiency, researchers have resorted to other approaches for developing combined constrained parity-check coding schemes [31,38]. The post-processing unit makes use of either Euclidean distance computations [32,37,40] or a bank of matched filters [35,38,53].

1.6 Motivation, Contribution and Organization of the Thesis

The brief overview presented in Sections 1.3 to 1.5 shows that the design of efficient and powerful codes is key to designing a high-performance PRML system. Therefore, our focus in this thesis is to design efficient constrained parity-check codes and effective post-processing techniques to improve the performance of PRML schemes for perpendicular recording systems. Two points are worth mentioning here. Firstly, as compared to longitudinal recording channels, perpendicular recording channels are characterized by different dominant error mechanisms at the output of the Viterbi detector. Therefore, the distance-enhancing constraints as well as the parity-check constraints needed to improve the performance in perpendicular systems may be quite different from that in the longitudinal case. Therefore, we investigate the design of new and efficient distance-enhancing codes and parity-check codes. Secondly, the post-processor design is based on the principle of the optimum receiver for multiple signal detection in communication theory. While existing parity-based post-processors are based on ML decision rule, it is expected that post-processors based on maximum a posteriori (MAP) decision rule should be superior. Therefore, we investigate MAP based post-processors and corresponding parity-check code design.

The thesis is organized as follows. Chapter 2 gives detailed description of the perpendicular recording system based on PRML detection scheme. Chapter 3 presents a brief survey of constrained modulation codes for PRML detection schemes. In particular, we review design techniques of codes that combine runlength and parity constraints. In Chapter 4, we present parity-check codes and parity-based post-processing techniques. A detailed analysis, not available in the literature, of the parity-based post-processors is also presented. Chapter 5 presents a novel constrained parity-check code with post-processing. This code, which combines of MTR runlength constraints and parity constraints, improves the bit-error rate (BER) performance of the Viterbi detector. In Chapter 6, we examine distance-enhancing constraints that are specifically matched to MAP-based post-processors. We propose a method for identifying constraints that optimize the performance of MAP-based post-processor. Eventually, Chapter 7 concludes the thesis and presents possible directions for further work.

Chapter 2

Perpendicular Magnetic Recording System

In this chapter, we set up the perpendicular magnetic recording system model which will be used throughout this thesis. In Section 2.1, we introduce a mathematical model for the perpendicular recording channel. In Section 2.2, we briefly review the principle of PRML detection starting from fundamentals. In Section 2.3, the principle of partial response (PR) equalization and the design of PR equalizer and target are presented. In Section 2.4, we describe the Viterbi algorithm starting from the principle of maximum likelihood sequence detection (MLSD). The performance analysis of Viterbi algorithm is also presented in this section.

2.1 Perpendicular Magnetic Recording Channel Model

In this section, starting from the block schematic given in Figure 1.1 of Chapter 1, we present the development of the discrete-time model of the perpendicular recording channel.

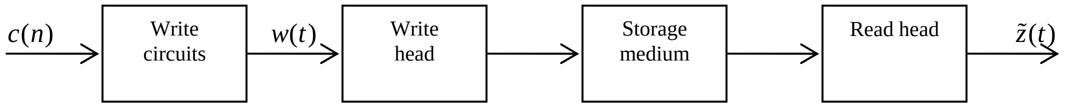

Figure 2.1: Block schematic of the recording channel of a digital magnetic recording system.

Shown in Figure 2.1 is part of Figure 1.1 from the input of the write circuits to the output of the read-head. By the term ‘recording channel’, we mean the cascade of the write circuits, write head, storage medium and read head.

The input $c(n)$ denotes the coded user data in NRZ format with $c(n) \in \{-1,1\}$. Here, ‘-1’ and ‘+1’ are equivalent to the NRZ bits ‘0’ and ‘1’, respectively. In other words, $c(n)$ denotes the polarity of the write current pulse for the n^{th} bit. The write circuits convert the coded bits sequence into write current pulses. If $p(t)$ denotes the write current pulse for a single bit, then the write current waveform can be written as

$$w(t) = \sum_n c(n) p(t - nT), \tag{2.1}$$

where T is the bit duration at the encoder output. The ideal $p(t)$ is a unit-amplitude rectangular pulse of duration T. That is, $p(t) = 1$ if $t \in [0,T]$ and $p(t) = 0$ if $t \notin [0,T]$. The write head converts the write-current waveform into magnetic flux that magnetizes the storage medium to store every bit as a small magnetized region on the disk. The read-head converts the magnetization on the disk into an electric signal. Let $f(t)$ denote the impulse response of the combination of the write-head, storage medium and read-head. Then, the reproduced voltage waveform at the read-head output can be written as [12]

$$\tilde{z}(t) = w(t) \otimes f(t) + \eta'(t) = \sum_n c(n) h(t - nT) + \eta'(t), \tag{2.2}$$

where $\eta'(t)$ represents the electronics noise picked up by the read-head, $h(t) = p(t) \otimes w(t)$, and ‘$\otimes$’ denotes the convolution operator. The electronics noise is modeled as white Gaussian with power spectral density $\frac{N_0}{2}$ Watts/Hz. Eq. (2.2) shows that the magnetic recording channel can be considered as a pulse-amplitude modulated system with input bits $c(n) \in \{-1,1\}$, bit response (or, symbol response) $h(t)$, and additive channel noise $\eta'(t)$. Since $h(t)$ is the output of the system for a single-bit input to the write circuits, it can be viewed as the output corresponding to the single pulse $p(t)$ input to the write-head. For this reason, $h(t)$ is also called the ‘pulse response’ of the recording channel. Noting that $p(t)$ can be expressed as $p(t) = u(t) - u(t-T)$ where $u(t)$ is the unit step function, we can express $h(t)$ as

$$h(t) = \big(u(t) - u(t-T)\big) \otimes f(t) = h_s(t) - h_s(t-T) \tag{2.3}$$

where $h_s(t) = u(t) \otimes f(t)$ is the step response of the recording channel. Therefore, $h(t)$ is also called the ‘dibit response’ of the recording channel since $p(t)$ contains two transitions spaced at T. If the input bits define an isolated transition, *i.e.* $....-1-1-1+1+1+1....$, then the output will be $2h_s(t)$. For this reason, $2h_s(t)$ is called the ‘transition response’ of the recording channel.

Based on experimental data, a pulse defined with a “tanh” hyperbolic function has been found to be a suitable model for the step response of perpendicular magnetic recording channels. This pulse is given by [55]

$$h_s(t) = \frac{A}{2} \tanh\left(\frac{\log(3)}{T_{50}} t \right), \tag{2.4}$$

where A is the pulse amplitude, and T_{50} is the time that $h_s(t)$ takes to rise from $-A/4$ to $+A/4$ (see Figure 2.2). For a given head/medium combination, T_{50} is an indicator of the extent of intersymbol interference (ISI) in the recording channel. It is also a measure of the density with which bits are written. Denoting the bit duration of user input data by T_u, the parameter defined by $D_u = T_{50} / T_u$ is called the user density, which is a measure of the recording density from the user's point of view. Traditionally, the data before encoding by ECC and modulation encoders (or, channel encoders) is called 'user data'

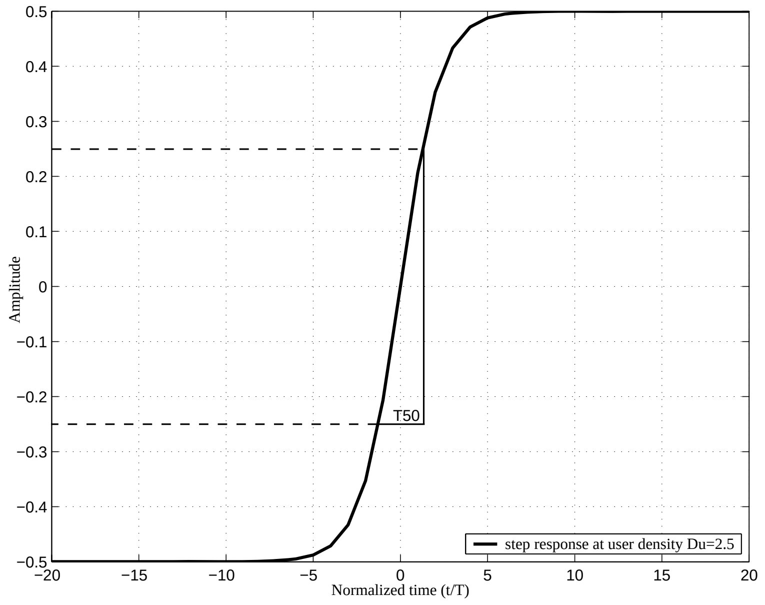

Figure 2.2: Step response of perpendicular magnetic recording channel model using 'tanh' pulse ($D_u = 2.5$, $R = 1$).

and the data after channel encoding is called 'channel data'. If R denotes the combined code rate of all the channel encoders, then the channel bit duration is given by $T = RT_u$. In our studies, the channel bit interval T is normalized to 1. Hence, T_{50} is given by $T_{50} = D_u T_u = D_u T / R = D_u / R$. We note here that the channel density $D_c = T_{50} / T = D_u / R$ increases as the code rate decreases. Figure 2.3 shows the bit response, $h(t) = h_s(t) - h_s(t - T)$, and the corresponding frequency response of 'tanh' perpendicular recording channels for linear recording densities $D_u = 2.0$ and $D_u = 3.0$.

Figure 2.3(a) shows that when the density D_u increases, the peak amplitude of the bit response decreases and its width increases. In other words, ISI increases and the energy of $h(t)$ decreases as the density increases. It is important to note that the bit response at user density $D_u = 3.0$ could be obtained from that at $D_u = 2.0$ by selecting a code rate $R = 2/3$. The observed degradation (*i.e.* increase in ISI and decrease in signal energy) is the manifestation of the code rate penalty, or the rate loss. Further, $T = RT_u$

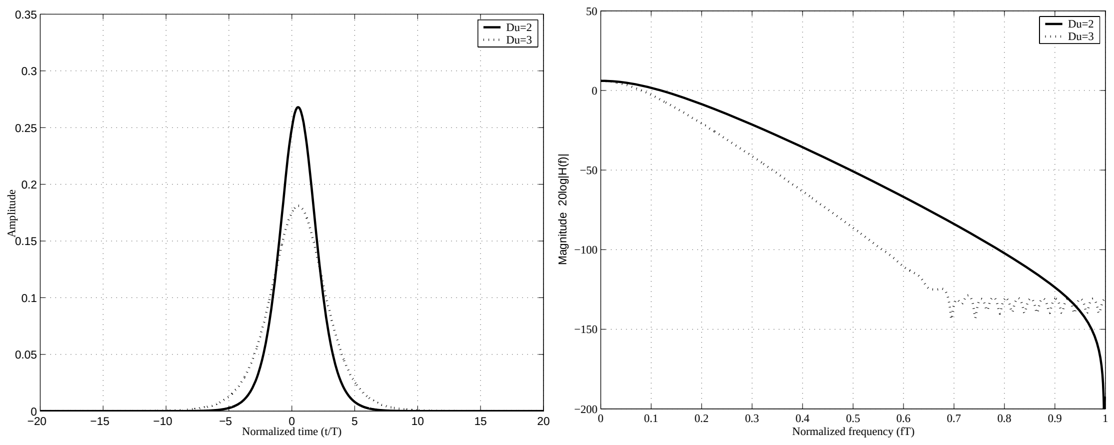

Figure 2.3: Channel responses for 'tanh' perpendicular channel with user densities Du=2.0 and Du=3.0. (a) Bit response, $h(t)$. (b) Frequency response, $H(f)$.

implies that the channel data rate is higher than the user data rate. In other words, noise power in the signal bandwidth $1/T$ increases with decrease in code rate. Consequently, we see a faster reduction in the SNR of the read-back signal with decreasing code rate. This is the reason why high-rate codes are highly desirable. Further, Figure 2.3(b) shows that the channel bandwidth decreases as the density increases, thereby necessitating the use of partial response equalization with controlled ISI instead of full-response equalization with zero ISI.

To develop a discrete-time model of the recording channel for our studies, we proceed as follows. Let B Hz be the bandwidth of the bit response $h(t)$. That is, the energy in the Fourier transform $H(f)$ of $h(t)$ for $|f| > B$ is negligible. Or, $|H(f)|^2 \approx 0$ for $|f| > B$. Consequently, we can sample the channel output at any rate exceeding $2B$ samples/second after limiting the noise bandwidth accordingly. Figure 2.3(b) shows that the channel bandwidth is between $\frac{1}{2T}$ and $\frac{1}{T}$ for typical recording densities. So, for convenience, we choose the low-pass filter bandwidth as $\frac{1}{T}$ and the sampling rate as $\frac{2}{T}$. Figure 2.4 shows the resulting sampling process. Here, L is an integer denoting the oversampling factor. In our case, $L = 2$.

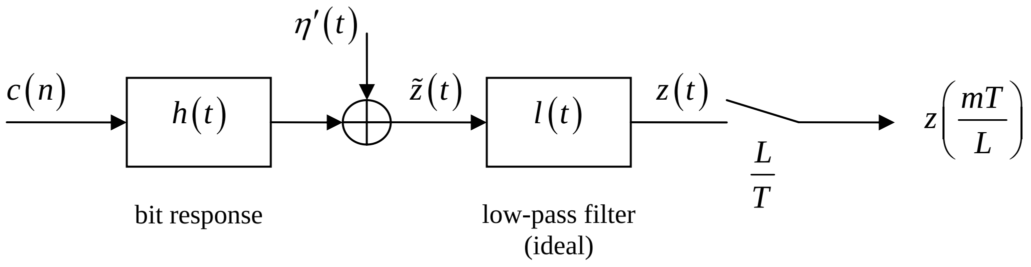

Figure 2.4: Sampling of the channel output.

From Figure 2.4, the sampled output of the channel can be given by

$$\begin{aligned} z(m) \triangleq z(t)\Big|_{t=\frac{mT}{L}} &= \sum_n c(n)h(t-nT) + \upsilon(t)\Bigg|_{t=\frac{mT}{L}} \\ &= \sum_n c(n)h\left((m-nL)\frac{T}{L}\right) + \upsilon\left(\frac{mT}{L}\right) \\ &= \sum_n c(n)h_{m-nL} + \upsilon(m) \end{aligned} \quad (2.5)$$

where $h_i \triangleq h(t)\big|_{t=\frac{iT}{L}}$ and $\upsilon(m) \triangleq \upsilon(t)\big|_{t=\frac{mT}{L}}$ with $\upsilon(t)$ denoting the low-pass filtered version of $\eta'(t)$. Because the low-pass filter is ideal with its bandwidth exceeding the bandwidth of $h(t)$, the low-pass filter does not affect the signal part of the sampled output. Further, it also follows that $\upsilon(m)$ is a discrete-time white Gaussian noise process with variance $\frac{N_0 L}{2T}$. Thus, Eq. (2.5) represents the sampled output of the recording channel and the resulting discrete-time model of the recording channel is shown in Figure 2.5.

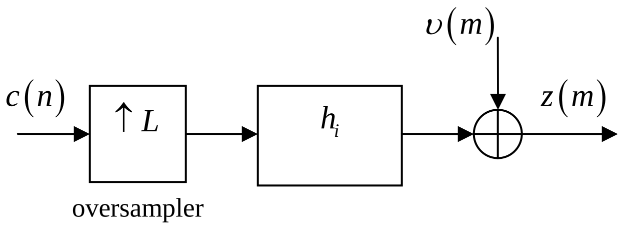

Figure 2.5: Discrete-time model of the recording channel.

We will be using this discrete-time model throughout this thesis.

2.2 PRML Principle

Forney [11] showed that the optimal ML receiver for a linear channel corrupted with ISI and additive white Gaussian noise (AWGN) includes a whitening matched filter (WMF) followed by a symbol-rate sampler (*i.e.* at rate $1/T$) and a maximum likelihood sequence detector. The symbol-rate sampled output of the whitening matched filter provides a set of sufficient statistics for optimal ML estimation of the input sequence. The ML sequence detector is implemented with a nonlinear recursive algorithm, called the Viterbi algorithm [46]. Even though the Viterbi-based implementation is computationally very efficient compared to brute-force search for the optimal sequence using a binary tree, the long channel memory of magnetic recording channels makes even the direct Viterbi-based approach extremely complex. The long channel memory arises from the fact that recording channels are highly frequency selective and band-limited with little energy near the band edge $1/2T$ (see Figure 2.3(b)). Further, the complexity of the Viterbi detector is exponential in the memory length of the channel. More details about Viterbi detection are available in Section 2.3. Some sub-optimal detection schemes, such as the fixed-delay tree search detection scheme, have been suggested to reduce the complexity of the detector without much loss in performance [12].

The principle of partial response (PR) equalization is an effective approach for shortening the channel memory of band-limited channels [56,12]. The idea is to find a PR signal that is spectrally similar to the recording channel $h(t)$ while the memory of the PR signal after sampling (at rate $1/T$) is much shorter than that of $h(t)$. Typical examples of PR signals (or, PR targets) used in magnetic recording are $1-D^2$ and $1+D-D^2-D^3$ for longitudinal and $1+2D+2D^2+D^3$ and $1+2D+3D^2+2D^3+D^4$ for perpendicular recording [12,45,48,14]. When an equalizer is used to shape the channel response into such short PR target responses, it is easily seen that the resulting complexity of the Viterbi detector is much reduced. Figure 2.6 illustrates a channel with PR equalizer. The equalizer w_i, which is a finite-impulse response filter, is designed to make the effective channel from $c(n)$ to $x(n)$ to be the selected PR target. Section 2.3 gives details on the

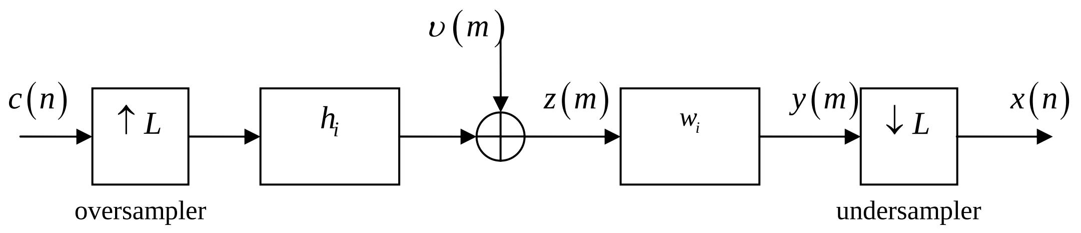

Figure 2.6: PR equalization of the recording channel.

design of the equalizer. Since it is seldom possible to find a short PR target that is perfectly identical to $h(t)$, the PR equalization process results in noise enhancement in certain frequency regions. Consequently, the Viterbi detector is no more optimum in the sense of MLSD. Therefore, choice of the PR target is key to good detection performance. While the PR target must be spectrally similar to the channel $h(t)$ so as to decrease noise enhancement, the length of the target must be small enough to decrease the complexity of the Viterbi detector.

The idea of applying PRML detection method, *i.e.* PR equalization followed by Viterbi detector, to magnetic recording channels dates back to the early 1970’s [13]. With the announcement by IBM on 1Gbits/in^2 demonstration in 1990, the PRML scheme rapidly became very popular. Currently, it is the most widely used detection technique in commercial disk drives.

2.3 Partial-Response Equalization

The perpetual push for higher user bit rates and higher storage densities results in a steady increase of linear ISI and noise disturbances in storage channels. Equalization, performed on the read-back signals, is used to compensate for ISI and noise distortions introduced by the recording channel on the data. Full-response equalization aims at canceling the ISI completely. However, this causes the noise to be enhanced seriously in the frequency regions where the magnitude of the channel frequency response is very small. Therefore, full-response equalization is not advisable for magnetic recording channels, since these are bandlimited channels. Consequently, as mentioned in Section 2.2, PR equalization is the approach used for shortening magnetic recording channels.

2.3.1 Design of PR Equalizer

We use a finite impulse response (FIR) PR equalizer to equalize the recording channel to a chosen PR target. If we choose the tap-spacing of the equalizer to be T, then its performance may depend heavily on the sampling phase of the channel output if the channel (*i.e.* $h(t)$) is not bandlimited to $\frac{1}{2T}$. Since the channel bandwidth depends on the recording density (see Figure 2.3(b)), we use the oversampled model of the recording channel shown in Figure 2.5 for our studies. Consequently, we need to design a fractionally-spaced equalizer (*i.e.* tap-spacing is T/L) as implied in Figure 2.6. Clearly, the fractional spacing allows the equalizer to be robust against variations in the sampling phase [57]. In order to design the equalizer tap weights, we consider the minimum mean-squared error (MMSE) approach. The MMSE approach aims to minimize both residual

ISI (*i.e.* mismatch of the equalized channel with the PR target) and additive noise in the channel. Because MMSE design considers the noise characteristics, MMSE equalizer minimizes noise enhancement and it exists even if channel response has spectral nulls [12].

Figure 2.7 shows the block schematic used for designing a T/L-spaced PR equalizer to equalize the recording channel h_i to the PR target g_k. The time indices n and m are associated with sequences which are at rates $\frac{1}{T}$ and $\frac{L}{T}$, respectively.

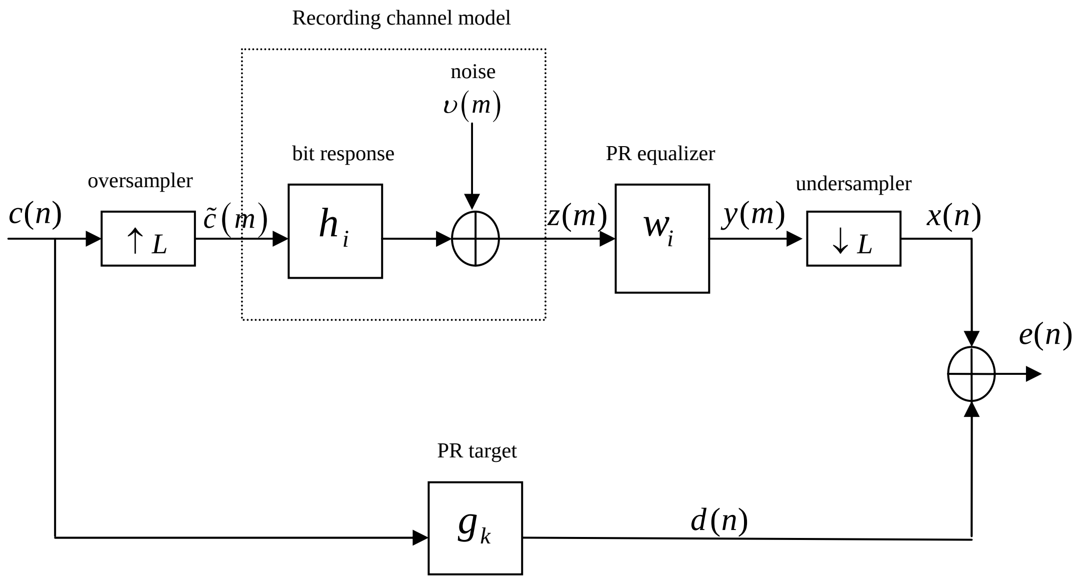

Figure 2.7: Block schematic for designing a T/L-spaced equalizer for equalizing the channel h_i to the PR target g_k.

Here, $\{c(n)\}$ represents the channel coded data sequence in NRZ ± 1 format, h_i is the sampled bit response of the recording channel, and w_i is the impulse response of the PR equalizer. The taps of h_i and w_i are at the spacing T/L where L is the oversampling factor. Further, $h_i = \tilde{h}_i - \tilde{h}_{i-L}$ where $\tilde{h}_i$ is the channel step response given by (see Eq. (2.4))

$$\tilde{h}_i = \frac{A}{2}\tanh\left(\frac{iR\log(3)}{LD_u}\right), \tag{2.6}$$

where R is the code rate of the channel encoder. The channel noise $\upsilon(m)$ is assumed to be white Gaussian and its variance, σ_υ^2, is determined from the channel SNR defined as

$$SNR(dB) = 10\log_{10}\frac{V_{op}^2}{\sigma_u^2}, \qquad \sigma_\upsilon^2 = \frac{\sigma_u^2 L}{R}, \tag{2.7}$$

where $\sigma_u^2 = \frac{N_0 R}{2T}$ is the variance of the noise in the user bandwidth $\frac{1}{T_u}$ and V_{op} is the base-to-peak value of the isolated transition response $2h_s(t)$.

The PR equalizer is designed using the MMSE criterion. The equalizer output, after down-sampling, is given by

$$x(n) = \underline{w}^T \underline{z}(nL + m_0), \tag{2.8}$$

where $\underline{w} = \left[w_0, ..., w_{N_w-1}\right]^T$ is the vector of the equalizer coefficients, N_w is the number of equalizer taps, $\underline{z}(m) = \left[z(m), ..., z(m - N_w + 1)\right]^T$ is the noisy channel output, m_0 is the sampling phase (*i.e.* the total delay from channel input to equalizer output), and the superscript 'T' denotes matrix transposition. Let the PR target be $\underline{g} = [g_0, ..., g_{N_g-1}]^T$, whose coefficients are T-spaced. Then, the desired signal at the output of the equalizer for instant n is given by

$$d(n) = \sum_{k=0}^{N_g-1} g_k c(n-k). \tag{2.9}$$

Therefore, the mean squared error at the equalizer output is given by

$$J\left(\underline{w}\right) = E\left[\left(x(n) - d(n)\right)^2\right], \tag{2.10}$$

which is equal to the sum of variances of the channel noise and residual ISI at the equalizer output. The optimum equalizer is obtained by minimizing $J\left(\underline{w}\right)$ with respect to $\underline{w}$ and the solution is given by the Wiener-Hopf equation

$$R_{zz}\underline{w}_{opt} = R_{zc}\,\underline{g}\,, \tag{2.11}$$

where $R_{zz} = E[\underline{z}(nL+m_0)\underline{z}^T(nL+m_0)]$ and $R_{zc} = E[\underline{z}(nL+m_0)\underline{c}^T(n)]$, with E[.] denoting the statistical expectation operator.

2.3.2 Design of PR Target

To optimize the performance of PRML systems, the PR target should be well designed to reduce mis-equalization and noise enhancement. Conventional PRML schemes employ standard targets with integer coefficients, which are chosen by examining their match to the actual channel response $h\left(t\right)$. The well-known example of standard targets for longitudinal recording is the Class 4 targets given by $\left(1-D\right)\left(1+D\right)^n$ where n is a positive integer [13,58]. Similarly, $\left[1,2,2,1\right]$ and $\left[1,2,3,2,1\right]$ are commonly used PR targets for perpendicular recording [14]. To examine which of these two perpendicular recording targets is more effective, we designed the equalizer for both cases, for an uncoded perpendicular recording channel (*i.e.* code rate $R=1$) with user density $D_u = 2$ and $SNR = 32$, and performed a spectral analysis. The results are shown in Figure 2.8. Observe that the magnitude responses of the equalizers show that noise enhancement is present in both cases. Moreover, in the case of the $[1,2,3,2,1]$ PR target, the shape of the magnitude response of the equalizer is flatter at low frequencies, compared to that for $[1,2,2,1]$. This reveals that, compared to $[1,2,2,1]$ PR target, $[1,2,3,2,1]$ PR target is spectrally more similar to the channel response. Consequently, the noise spectrum at the Viterbi detector (VD) input can be expected to be flatter for $[1,2,3,2,1]$ as compared to

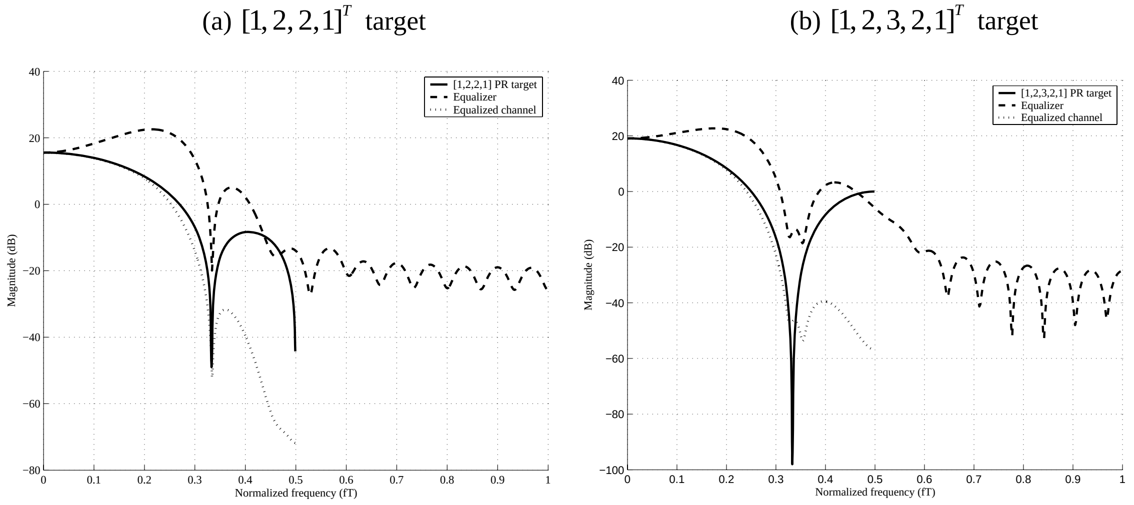

Figure 2.8: Magnitude responses of the equalizer, PR target and equalized channel for (a) PR target $[1,2,2,1]$ and (b) PR target $[1,2,3,2,1]$.

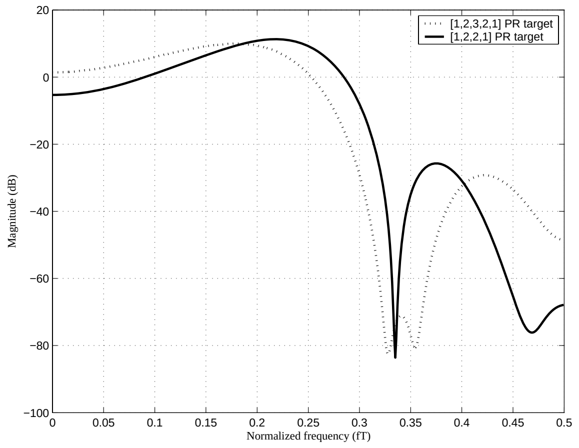

Figure 2.9: Normalized power spectral density of equalized noise at VD input for the PR targets $[1,2,2,1]$ and $[1,2,3,2,1]$.

$\left[1,2,2,1\right]$, as can be seen from Figure 2.9. Since optimality of VD requires the noise to be white and Gaussian, we can expect that VD is more optimal with $\left[1,2,3,2,1\right]$ as compared to $\left[1,2,2,1\right]$. For this reason, in our studies with integer-valued PR targets, we choose $\left[1,2,3,2,1\right]$ as our target.

Due to the integer constraint, the standard targets do not provide close spectral match to the natural channel responses especially at high densities, and thus result in substantial noise enhancement. Instead, the generalized PR (GPR) targets with real-valued coefficients can provide better match, and consequently, achieve significant performance gain. The most widely used method for GPR target design is to jointly design the equalizer and target with the MMSE criterion (see Eq. (2.10)) [15,59,60,61]. These are constrained optimization approaches. Another approach to design GPR targets is by minimizing the dominant error event probability in the Viterbi detector [32,36,15]. But, this approach is computationally very costly since numerical searches are required to find the solution as analytical solutions are not available. However, as reported in [15,61,62], the MMSE approach with monic constraint (*i.e.* first coefficient of the GPR target is constrained to be unity) has been found to result in solutions that are near-optimal in the sense of minimizing the dominant error event probability. Therefore, in this thesis, we use the monic-constrained MMSE approach for designing GPR targets. Finally, we may remark that the decision feedback equalization (DFE) system [16] and noise-predictive maximum-likelihood (NPML) system [5] can be viewed as special cases of PRML with GPR target.

2.4 Sequence detection

The Viterbi algorithm is a computationally efficient implementation of MLSD if the channel noise is white Gaussian at the detector input. In this section, we briefly review the Viterbi algorithm and its performance analysis.

2.4.1 Viterbi Algorithm

From Figure 2.6, the equalizer output (or, detector input) can be written as

$$x(n) = \sum_{k=0}^{Ng-1} g_k c\left(n-k\right) + \eta\left(n\right) \tag{2.12}$$

where $\eta\left(n\right)$ represents the sum of the residual ISI and the equalized channel noise. For simplicity, the noise $\eta\left(n\right)$ at the detector input is assumed in this section to be white and Gaussian with variance σ_η^2. The MLSD obtains the detected bits sequence $\underline{\hat{c}} = \left[\hat{c}\left(0\right),...,\hat{c}\left(N-1\right)\right]^T$ by maximizing the joint probability density function (pdf) of the received samples $\underline{x} = \left[x\left(0\right),...,x\left(N+N_g-2\right)\right]^T$ conditioned on the input bits sequence $\underline{c} = \left[c\left(0\right),...,c\left(N-1\right)\right]^T$ [11]. In other words, the decision rule of the MLSD is

$$\underline{\hat{c}} = \arg\max_{\underline{c}_i} p_{\underline{x}}\left(\underline{x}/\underline{c}_i\right) \tag{2.13}$$

where 'arg' refers to the maximizer of the joint pdf $p_{\underline{x}}\left(.\right)$ and $\underline{c}_i$, $i = 1,..,2^N$, is one of the 2^N possible input bit sequence $\underline{c}$.

For a given input sequence $\underline{c}$, each sample $x(n)$ is a random variable which depends only on the noise sample $\eta\left(n\right)$. Since the noise sequence is assumed to be white

Gaussian, the sequence $\underline{x}=\left\{x(0),x(1),...,x\left(N+N_g-2\right)\right\}$ comprises uncorrelated Gaussian random variables. The pdf of $\underline{x}$ conditioned on $\underline{c}$ is then given by

$$p_{\underline{x}}\left(\underline{x}/\underline{c}\right)=\prod_{n=0}^{N+N_g-2} p_\eta\left(x(n)/\underline{c}\right)=\prod_{n=0}^{N+N_g-2}\frac{1}{\sqrt{2\pi}\sigma_\eta}\exp\left(-\frac{\left[x(n)-x_c(n)\right]^2}{2\sigma_\eta^2}\right) \tag{2.14}$$

where $p_\eta\left(v\right)=\frac{1}{\sqrt{2\pi}\sigma_\eta}\exp\left(-\frac{v^2}{2\sigma_\eta^2}\right)$ is the pdf of $\eta\left(n\right)$, and $x_c(n)=\sum_{k=0}^{Ng-1} g_k c\left(n-k\right)$ is the reconstruction of the signal part of $x(n)$ based on the assumed input data $\underline{c}$. Since σ_η^2 is a constant, maximizing $p_x\left(\underline{x}|\underline{c}\right)$ is equivalent to minimizing the Euclidean distance

$$J\left(\underline{c}\right)=\sum_{n=0}^{N+N_g-2}\left[x(n)-x_c(n)\right]^2=\left\|\underline{x}-\underline{x}_c\right\|^2 \tag{2.15}$$

Thus, the MLSD aims at selecting the vector $\underline{c}$ that minimizes the Euclidean distance $J\left(\underline{c}\right)$. However, to implement the MLSD, 2^N Euclidean distances $J\left(\underline{c}_i\right)$ need to be computed. The underlying computational requirement grows exponentially with N and becomes huge for reasonably large values of N. Direct implementation of the MLSD is consequently impossible for reasonable N.

The Viterbi algorithm (VA) is a clever implementation of the MLSD with emphasis on reducing the computational complexity [46]. A concise and convenient structure for representing the input data is the trellis shown in Figure 2.10, where the '+' and '-' along any branch of the trellis represents the bit $+1$ or -1 associated with the path that passes through that branch. To represent the 2^N possible input sequences $\underline{c}$, the trellis requires only N stages. As n increases, each stage of the trellis shows the progress of all the input data paths under consideration.

At each stage n of the trellis and for each state $S_k(n)$, the Viterbi detector remembers the data path, called the survivor path, that has minimum Euclidean distance, called the survivor the path metric, among the paths ending at state $S_k(n)$. In other words, half of the paths over which we need to compute the path metric are dropped at each stage of the trellis. After all stages are completed, the survivor path that has the minimum survivor

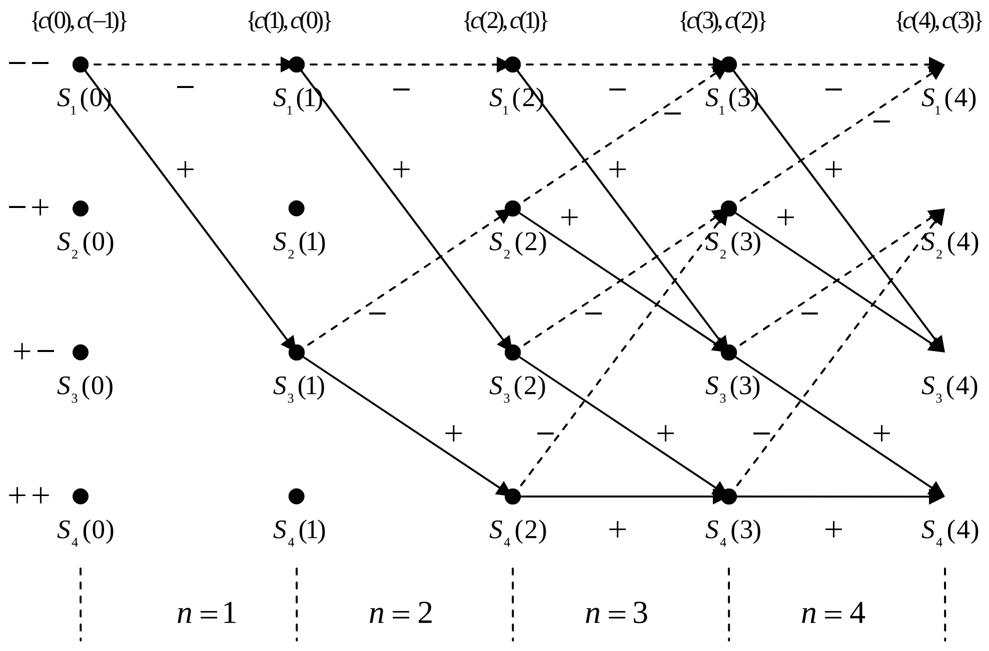
{c(0), c(−1)}
{c(1), c(0)}
{c(2), c(1)}
{c(3), c(2)}
{c(4), c(3)}
−−
−+
+−
++
S_1(0) S_1(1) S_1(2) S_1(3) S_1(4)
S_2(0) S_2(1) S_2(2) S_2(3) S_2(4)
S_3(0) S_3(1) S_3(2) S_3(3) S_3(4)
S_4(0) S_4(1) S_4(2) S_4(3) S_4(4)
+
−
n = 1
n = 2
n = 3
n = 4

Figure 2.10: Trellis structure for channel with memory $N_g - 1 = 2$.

path metric is selected as the detected path. In practice, the survivor paths corresponding to the states $S_k(n)$, $k = 1,...,2^{N_g-1}$, would have converged to a single path for time instants less than or equal to $n - K(N_g - 1)$ for a sufficiently large positive integer K. Usually, K is chosen in the range of 5 to 10. The delay with which the detector makes decisions on the bits is thus $K(N_g - 1)$ and is called the detection delay. This helps to reduce the memory requirements of the Viterbi detector.

2.4.2 Performance Analysis of Viterbi Detector

The bit error rate (BER) performance of VD can be estimated using simulations as

$$P_b = \frac{\text{number of erroneous bits}}{\text{number of recorded bits}} = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=0}^{N-1} \left| c(n) - \hat{c}(n) \right| . \tag{2.16}$$

In order to simulate scenarios with BER of 10^{-6} or less, we will need to run the simulations over hundreds and thousands of millions of bits and it can be very time-consuming to do this. An alternative way of estimating the performance stems from the concept of error events. For a pair of input data vectors $\underline{c}$ and $\underline{c}'$, let us define the error sequence $\underline{c} - \underline{\hat{c}}$. This error sequence defines an error event if there exists two integers $k_1 \le k_2$ such that $e(k) = c(k) - \hat{c}(k) = 0$ for $k < k_1$ and $k > k_2$, $e(k_1) \neq 0$, $e(k_2) \neq 0$ and any run of zeros in the subsequence $[e(k_1+1),...,e(k_2-1)]$ is smaller than the channel memory[1] $N_g - 1$. Then, $\underline{e} = \left[e(k_1), e(k_1+1), ..., e(k_2) \right]^T$ defines an error event of length $k_2 - k_1 + 1$. In the trellis, the error events are seen as distinct separations between the actual state sequence and the detected state sequence. The Viterbi detector then produces an error when the detected trellis path differs from the correct path by a sequence of error events. The union bound[2] provides an upper bound to the probability that an error event starts at some time k_1 [57,12]

[1] When the noise at the VD input is correlated, the maximum run of zeros in the subsequence $[e(k_1+1),...,e(k_2-1)]$ can be bigger than $N_g - 1$.

[2] Appendix C provides a detailed analysis on the performance of Viterbi detector.

$$P_{event} \leq \sum_{e \in E} \mathrm{P}_e \tag{2.17}$$

where E is the set of error events at the detector output, $\mathrm{P}_e = \sum_{\underline{c} \in C_e} \Pr\left[\underline{\hat{c}} / \underline{c}\right] \Pr\left[\underline{c}\right]$ is the probability that the error event $\underline{e}$ starts at some time k_1, and C_e is the set of data patterns $\underline{c}$ which support the error event $\underline{e}$ starting at time k_1. By taking into account the noise correlation at the VD input and the residual ISI, the probability $\Pr\left[\underline{\hat{c}} / \underline{c}\right]$ of detecting $\underline{\hat{c}}$ instead of $\underline{c}$ is given by [63]

$$\Pr\left[\underline{\hat{c}} / \underline{c}\right] = Q\left(\frac{\left\| \underline{e}_g \right\|^2 + 2\underline{e}_g{}^T \left(\underline{x} - \underline{c} \otimes \underline{g} \right)}{2\sqrt{\underline{e}_g{}^T R_{\eta\eta} \underline{e}_g}} \right) \tag{2.18}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-t^2/2} dt$, $\underline{e}_g = \underline{e} \otimes \underline{g}$, and $R_{\eta\eta}$ is the autocorrelation matrix of the equalized channel noise at the detector input. Note that the conditional probability given in Eq. (2.18) depends on the actual data pattern $\underline{c}$. Using the union bound, the error event probability is given by

$$P_{event} \leq \sum_{\underline{e} \in E} \sum_{\underline{c} \in C_e} Q\left(\frac{\left\| \underline{e}_g \right\|^2 + 2\underline{e}_g^T \left(\underline{x} - \underline{c} \otimes \underline{g} \right)}{2\sqrt{\underline{e}_g^T R_{\eta\eta} \underline{e}_g}} \right) \Pr\left[\underline{c}\right], \tag{2.19}$$

where $\Pr\left[\underline{c}\right]$ is the probability of the data pattern $\underline{c}$. Therefore, the BER can be upper-bounded as

$$P_b \leq \sum_{\underline{e} \in E} \sum_{\underline{c} \in C_e} Q\left(\frac{\left\| \underline{e}_g \right\|^2 + 2\underline{e}_g^T \left(\underline{x} - \underline{c} \otimes \underline{g} \right)}{2\sqrt{\underline{e}_g^T R_{\eta\eta} \underline{e}_g}} \right) \Pr\left[\underline{c}\right] w(e), \tag{2.20}$$

where $w(e)$ is the Hamming weight of the error event $\underline{e}$.

A measure of practical interest is known as the effective detection SNR [63] and is given by

$$SNR_{eff} = \min_{\underline{c},\hat{\underline{c}}} \left(\frac{\left\| \underline{e}_g \right\|^2 + 2\underline{e}_g{}^T \left(\underline{x} - \underline{c} \otimes \underline{g} \right)}{2\sqrt{\underline{e}_g{}^T R_{\eta\eta} \underline{e}_g}} \right)^2 . \tag{2.21}$$

At moderate to high SNR, the performance of the system is dominated by the error events associated with SNR_{eff} . Therefore, the BER can be approximated as

$$P_b \approx \sum_{\underline{e} \in E_{\min}} Q\left(\sqrt{SNR_{eff}} \right) \pi(e) w(e), \tag{2.22}$$

where $E_{\min}$ is the set of error events associated with SNR_{eff} and $\pi(e)$ is the sum of the probabilities $\Pr[\underline{c}]$ of all data patterns $\underline{c}$ that support the error events in $E_{\min}$. There is, in practice, a unique error event that is associated with SNR_{eff} .

For a more accurate analysis of the BER performance of the VD, the effective distances associated with error events is defined :

$$d_{eff}(e) \triangleq \min_{\underline{c} \in C_e} \left(\frac{\left\| \underline{e}_g \right\|^2 + 2\underline{e}_g^T \left(\underline{x} - \underline{c} \otimes \underline{g} \right)}{2\sqrt{\underline{e}_g^T R_{\eta\eta} \underline{e}_g}} \right). \tag{2.23}$$

Obviously, the error events with minimum effective detection SNR correspond to the error events associated with minimum effective distance. The effective distances $d_{eff}(e)$ helps to identify the dominant error events and to estimate their respective probabilities. Under some channel conditions (*e.g.* high densities), there may be more than one set of dominant error events with their corresponding effective distances being very close to each other [86].

2.5 Summary

This chapter has reviewed the principle of PRML detection and its application to perpendicular recording channels. We have shown that the Viterbi detection in PRML is not optimal in the ML sense, because of the correlation of the noise at its input. Based on the analysis of the correlation of the noise at the VD input, we have also shown that $[1,2,3,2,1]$ is a suitable PR target that minimizes the noise correlation and results in a VD with acceptable computational complexity. Finally, we have given expressions for estimating the BER performance of the VD under various channel conditions.

Chapter 3

Constrained Codes for PRML Receivers

In this chapter, we elaborate on the topic of constrained modulation codes and their application to improve detection performance in PRML receivers. Our focus, in particular, is to review the different approaches to design efficient constrained codes that result in performance gain by prohibiting certain specified differences between constrained sequences. Since different distance-enhancing constraints with different capacity may eliminate the same dominant error event, finding high capacity constraints is very important in the design of distance-enhancing codes [17]. Distance-enhancing codes, which use strong constraints to eliminate certain data sequences, are presented in Section 3.1. Use of relaxed (weak) constraints supports the design of higher rate codes. Hence, in some cases, the weakly constrained codes may outperform strongly constrained codes [18]. A survey of high rate modulation codes, including weakly constrained codes, is presented in Section 3.2. Since high code rates and low decoding complexity are highly attractive, there have also been investigations to combine parity-check constraints and modulation constraints. Various techniques for equipping the modulation code with error control capabilities are presented in Section 3.3.

Since high code rates are far too important in current recording systems, we will not be discussing in this chapter on conventional (d, k) codes, which were used in the early days of recording systems, since these codes usually have low code rates (*e.g.* rate $2/3$ $(1,7)$ code, rate $1/2$ $(1,3)$ code). References [51] and [26] provide good reviews on these codes.

The original contributions in this chapter are as follows. We give an analytical investigation of the effect of code rate in perpendicular recording channels.

3.1 Distance-Enhancing Codes

In this section, we discuss the various aspects related to the design of efficient distance-enhancing codes.

3.1.1 Constraint Design Strategy

As mentioned already, the purpose of distance-enhancing codes is to improve detection performance by prohibiting the data patterns that support the dominant error mechanisms in the detector. Consequently, the first step in the design of distance-enhancing codes is identification of the dominant error mechanisms. Error event characterization of the detector, *i.e.* examining the probabilities of the various error events at the detector output, provides this information. Recall from Chapter 2 (Section 2.4.2) that associated with each error event $\underline{e}$ is an effective distance $d_{eff}\left(e\right)$ (see Eq. (2.23), Chapter 2) and the probability of the error event can be decreased by increasing this distance. Further, according to Eq. (2.22), the bit error rate (BER) can be reduced by enhancing the effective detection SNR, denoted SNR_{eff} , which is the minimum of all the squared effective distances associated with all the error events. However, the distance gain is achieved at the cost of decrease in coding efficiency, *i.e.* loss in code rate. To see this, we recall the expression for SNR_{eff} from Chapter 2. Neglecting the effect of mis-equalization, we get (from Eq. (2.21))

$$SNR_{eff} = \min_{\underline{e} \in E} \frac{\left\| \underline{e}_g \right\|^4}{4 \underline{e}_g^T R_{\eta\eta} \underline{e}_g}, \tag{3.1}$$

where E denotes the set of all error events $\underline{e}$, $\underline{e}_g = \underline{e} \otimes \underline{g}$, $R_{\eta\eta}$ is the autocorrelation matrix of the noise at the detector input, and $\underline{g}$ is the PR target. Recall also from Section 2.1 of Chapter 2 that the use of a code with rate R causes i) increase in the variance of the channel noise by a factor $1/R$ and ii) decrease in the signal energy (*i.e.* $\int h^2(t)\,dt$ or $\sum_i h_i^2$) due to increase in channel density. Clearly, these effects reflect in decreasing the SNR_{eff} given in (3.1) [22, 35]. This is why high rate distance-enhancing codes are particularly desirable. The overall performance gain is known as the coding gain, which is generally expressed as the saving in the SNR provided by the coded scheme as compared to the uncoded scheme, to achieve a specified BER performance.

The design of distance-enhancing constrained codes involves the following steps:

(a) Error event characterization for the given channel and identification of the dominant error events.

(b) Determination of a list of error strings that will prevent the occurrence of error events identified in Step (a).

(c) Identification of constraints which will prevent the error strings in Step (b) from occurring, and tuning of the Viterbi detector to the identified constraints.

(d) Construction of an efficient code.

Step (a) can be performed using simulations by collecting error events at the output of the Viterbi detector over a long sequence of data bits. This can be quite time consuming. Another approach is to use error event characterization algorithms such as those presented in [19, 64]. Yet another approach is to do a search by computing the effective distance for a reasonably comprehensive list of error events. Step (b) is usually straight-forward, when the number of error events in consideration is small. For Step (c), an enumeration search is used to find suitable constraints. It is highly desired that the selected constraints produce a good and practical distance-enhancing code, *i.e.* a code with good coding gain, high rate, simple encoder and decoder, and low-complexity detector. The problem of finding such interesting constraints is still an open problem. Bounds on the Shannon capacity of such constraints (*i.e.* a measure of the maximum efficiency with which a code can implement a constraint) are presented in [17]. The capacity, denoted *Cap*, of distance-enhancing constraints usually satisfies $Cap < 0.9$. For this range of capacities, a powerful tool for designing the constrained code in Step (d) is given by the 'Adler Coppersmith Hassner' (ACH) algorithm [20], which is also known as the 'state-splitting algorithm'. Alternatively, constrained block codes can be designed through computer search [8]. However, this method may result in codes with lower code rates.

3.1.2 Identification of High Capacity Distance-Enhancing Constraints

The maximum transition run (MTR) codes represent a class of distance-enhancing codes that are designed according to the above described design strategy. These codes were initially developed for high-density PRML channels with dominant error event $\{+2,-2,+2\}$ [7,8,9,21]. This error event corresponds to the error sequence $[0...0\ 2\ \text{-}2\ 2\ 0...0]$, where the number of zeros preceding and following [2 -2 2] is at least equal to the channel memory.

For the sake of convenience, we call the sequence obtained by prefixing and/or postfixing the error event by one or two zeros as an 'error string' (*e.g.* $\left[0\ \ +2\ \ -2\ \ 2\right]$, $\left[+2\ \ -2\ \ 2\ \ 0\right]$, $\left[0\ \ +2\ \ -2\ \ 2\ \ 0\right]$, *etc.*). If any of the error string is eliminated by a code constraint, it would prevent the occurrence of the underlying error event. For example, the pairs of data patterns supporting the error string [0 2 -2 2] of the error event $\{+2,-2,+2\}$ are given in Table 3.1. In each of the two pairs, at least one of the sequences

Table 3.1: Data patterns in NRZ {-1,1} format supporting [0 2 -2 2] error string.

CASE 1	-1	1	-1	1	3 consecutive transitions
	-1	-1	1	-1	2 consecutive transitions
CASE 2	1	1	-1	1	2 consecutive transitions
	1	-1	1	-1	3 consecutive transitions

contains 3 or more consecutive transitions. Therefore, the error event $\{+2,-2,+2\}$ can be eliminated by allowing at most 2 consecutive transitions in the write current sequences, or equivalently at most 2 consecutive '1's in the input data sequence in NRZI $\{0,1\}$ format. Such constraints are known as MTR constraints and are denoted $MTR(j)$ to emphasize the number of allowed consecutive transitions. The set of constrained sequences satisfying the $MTR\left(j=2\right)$ constraint, which is required to eliminate the $\left\{+2,-2,+2\right\}$ error event, are obtained by reading off the labels of any path from the state transition diagram shown in Figure 3.1. The construction of similar transition diagrams representing general runlength constraints will be presented in Section 6.2.1. The labels on the branches of the state transition diagram give the NRZI $\left\{0,1\right\}$ bits.

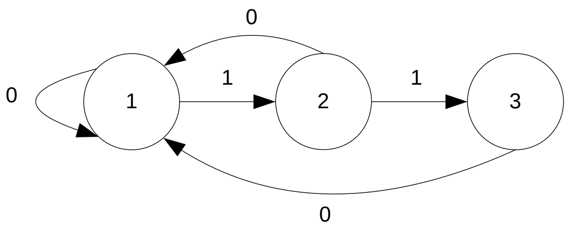

Figure 3.1: State transition diagram for $MTR\left(j=2\right)$ constraint.

The capacity of the $MTR(j=2)$ constraint is the highest achievable code rate and is defined by [9]

$$Cap = \log_2(\lambda_{\max}) \tag{3.2}$$

where $\lambda_{\max}$ is the largest real eigenvalue of the state transition matrix

$$A=\begin{bmatrix}1 & 1 & 0\\ 1 & 0 & 1\\ 1 & 0 & 0\end{bmatrix},$$

where for $i,j=1,2,3$, $A(i,j)=1$ if there is an edge from state i to state j and $A(i,j)=0$ otherwise. For the $MTR(j=2)$ constraint, with capacity 0.8791, a rate 6/7 state-dependent code [65] and a rate 16/19 block-code [8] have been designed. These codes include the k-constraint also, in addition to the $MTR(j=2)$ constraint. Since k-constraint is important in recording systems, MTR codes are usually denoted

'$MTR(j,k)$', thereby indicating the MTR and k constraints. As mentioned earlier, the d-constraint in such codes is zero.

In order to generate higher rate distance-enhancing codes, Bliss [21] and Fitzpatrick and Modlin [9] proposed time-varying MTR constraints. To identify the time-varying constraint for the error event $\{+2,-2,+2\}$, the error string [0 2 -2 2 0] and the 4 data patterns supporting this error string are considered, as shown in Table 3.2. Any code

Table 3.2: Data patterns in NRZ {-1,1} format supporting [0 2 -2 2 0] error string.

CASE 1	-1	1	-1	1	-1	4 consecutive transitions
	-1	-1	1	-1	-1	2 consecutive transitions
CASE 2	-1	1	-1	**1**	1	3 transitions end at time j
	-1	-1	1	-1	**1**	3 transitions end at time $j+1$
CASE 3	1	1	-1	1	**-1**	3 transitions end at time $j+1$
	1	-1	1	**-1**	-1	3 transitions end at time j
CASE 4	1	1	-1	1	1	2 consecutive transitions
	1	-1	1	-1	1	4 consecutive transitions

that allows 3 consecutive transitions to end at only on odd numbered bits or only on even numbered bits eliminates the $\{+2,-2,+2\}$ error event. This constraint is referred as a 'modulo 2 time-varying MTR constraint'. The transition diagram for this constraint is shown in Figure 3.2, where "square" and "circle" states correspond to even and odd time indices.

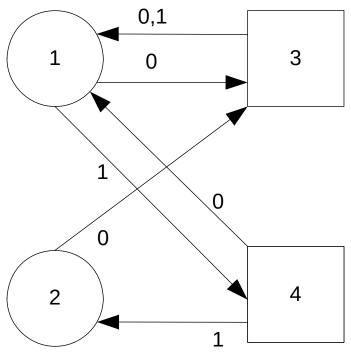

Figure 3.2: State transition diagram for the modulo 2 time-varying MTR constraint.

The transition matrix associated with this diagram is

$$A = \begin{bmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 2 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \end{bmatrix}.$$

Therefore, the Shannon capacity for the modulo 2 time-varying MTR constraint is 0.9163. The construction of rate 8/9 block codes which satisfy this constraint are presented in [9, 21]. Later, Nikolic *et al.* [66] proposed a rate 8/9 time-invariant $MTR(j = 2)$ code, thereby making detector implementation simpler.

An effective method of generating even higher rate distance-enhancing codes, referred to as forbidden list codes, was introduced by Karabed *et al.* [54]. These codes are designed so that if two sequences are separated by a dominant error event, then one or both of the sequences are forbidden. The error strings used in this design will have more preceding and/or following zeros [9]. For $\{+2, -2, +2\}$ error event, the error string [0 0 2 -2 2 0 0] was considered. Similar to the previous tables, we can give the data patterns that support [0 0 2 -2 2 0 0] error string. Among the resulting 16 pairs of data patterns, only two pairs do not contain at least one pattern with 4 consecutive transitions, as shown in Table 3.3. The forbidden list constraint $\mathcal{F} = \{1111, 001110\}_{NRZI}$ which forbids

Table 3.3: Pair of data patterns that support the error string [0 0 2 -2 2 0 0] while containing not more than 3 consecutive transitions.

	Patterns in NRZ format	Patterns in NRZI format
CASE 4	-1 -1 1 -1 1 1 1	0 1 1 1 0 0
	-1 -1 -1 1 -1 1 1	0 0 1 1 1 0
CASE 5	1 1 1 -1 1 -1 -1	0 0 1 1 1 0
	1 1 -1 1 -1 -1 -1	0 1 1 1 0 0

NRZI strings '1111' and '001110' eliminates the error event $\{+2,-2,+2\}$. This constraint has capacity 0.9132. A detailed treatment of forbidden list codes given in Chapter 6 shows how this capacity can be computed. By considering more zero symbols preceding and following the error event [9], one can show that the forbidden list constraint $\mathcal{F}=\{1111,100(00)1110\}_{NRZI}$, where the notation (00) means that 00 can be repeated any number of times, eliminates the error event $\{+2,-2,+2\}$. This constraint has capacity 0.925 and supports codes with rate as high as 12/13. A rate 9/10 block code was presented in [9].

3.1.3 Encoder/Decoder Design

As mentioned in the previous section, there is no explicit rule for designing a code once a constraint is given. The rate of distance-enhancing codes usually advises code designers to opt for block codes. When the block length is not too large, the encoder and decoder can be practically implemented with look-up tables. Block codes can be viewed as state-dependent codes with a single state. Because block codes represent a subclass of state-dependent codes, it is clear that they are less efficient than state-dependent codes with multiple states. The efficiency is measured by the ratio between the code rate R and the capacity Cap of the constraint, $\eta = R/Cap$. As an example, the rate 9/10 block code presented in [9] is 97.30% efficient. The efficiency can be improved by designing codes with the help of finite-state coding theory [10], which proves the existence of a state-dependent code with code rate R, as long as $R < Cap$. As a result, the constraint of the

9/10 block code [9], with capacity 0.925, makes it possible to design a 99.79% efficient rate 12/13 state-dependent code.

The design of block codes can be done by using computer search to generate all possible sequences of a given length q which can be freely concatenated without violating the specified constraints [8,22,23]. A few methods have been proposed for selecting the best codewords among the eligible sequences of length q and for mapping the source words to the selected codewords [24,25]. There is no general method, however. In every design, special attention should be given to the leading and trailing bits of eligible sequences and to the required k-constraint. In order to get a freely concatenable set of codewords, constraints are imposed on the leading and trailing bits. For instance, concatenated NRZI sequences of a given length that do not start or end with NRZI '11' string satisfy the $MTR(j = 2)$ constraint at the boundary [8]. The k-constraint is superimposed on the given constraints to ease the timing recovery process.

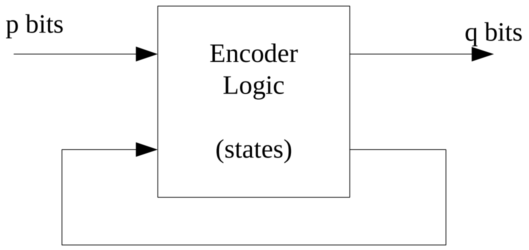

Figure 3.3: Finite-state encoder.

The existence of finite-state codes, pictured in Figure 3.3, with code rate less than the constraint capacity is guaranteed by the finite-state coding theorem [26]. The ACH algorithm (also called state-splitting algorithm) is a powerful algorithm for designing such codes [20]. In Appendix A, the state-splitting algorithm is presented and applied to a relevant example for this thesis. Finite-state codes usually require much more complex encoder and decoder than block codes. Capacity approaching finite-state codes with few states for simple implementation and simple decoders that avoid or minimize error propagation are highly desirable. Research in the design of efficient finite-state codes has mainly focused on finding codes with relatively simple decoders. The finite-state codes obtained by applying the state-splitting algorithm are sliding-block decodable [10, 67]. In other words, decoders for such codes need to look at the preceding and following codewords in order to decode the current codeword, as shown in Figure 3.4. Sliding-block decoders limit the error propagation to a finite number of codewords.

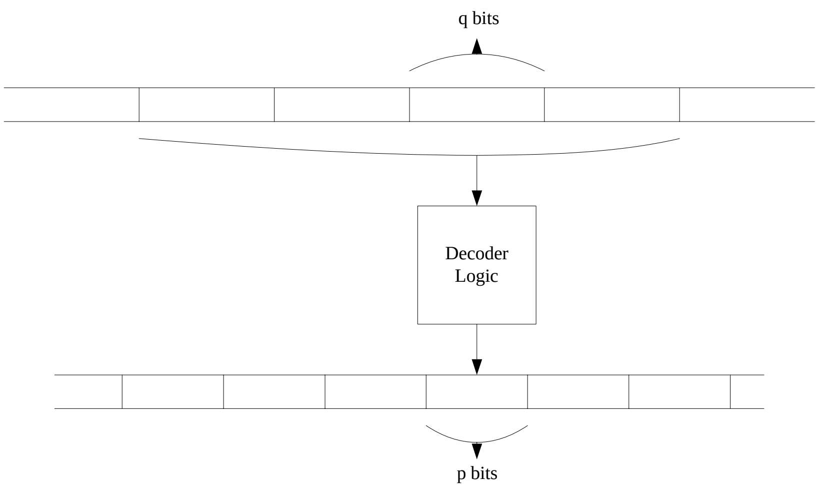

Figure 3.4: Sliding-block decoder.

A popular subclass of sliding-block decoders are block decoders which can make decision on a received source word with the use of a single codeword. Encoders associated with such decoders are called block-decodable encoders [26]. Recently, Chaichanavong and Marcus [71] presented methods for designing optimum block-decodable encoders with maximum code rate. The methods presented in [71], for instance, as well as much of the current research on finite-state coding are contributing to the results of the pioneering work done by Franaszek [72].

3.2 High-Rate Modulation Codes

In order to motivate the need for the development of codes with very high code rates, we shall first examine how code rate affects the SNR at the channel output. Thereafter, we shall discuss high rate weakly constrained codes.

3.2.1 Effect of Code Rate[3]

As compared to uncoded systems, coded systems suffer from the increased difficulty to equalize the read-back signal to the PR target at higher channel densities. As the channel density increases, the channel frequency response becomes less similar to the frequency response of the PR target, unless we are willing to increase the length of the target. This results in noise enhancement at the equalizer output.

In order to illustrate the effect of code rate, we shall evaluate the SNR at the output of the magnetic recording channel, shown in Figure[4] 3.5. For the sake of convenience, we assume $L=1$ in this section.

[3] Moon [70] has analytically investigated the effect of code rate on SNR in longitudinal recording channels. However, such an analysis is not available in the literature for perpendicular recording channels. Further, the ‘tanh’ model of the perpendicular recording channel makes the analysis rather difficult, as compared to the Lorentzian model used in the longitudinal case.

[4] This figure is the same as Figure 2.5 and is repeated for the sake of convenience.

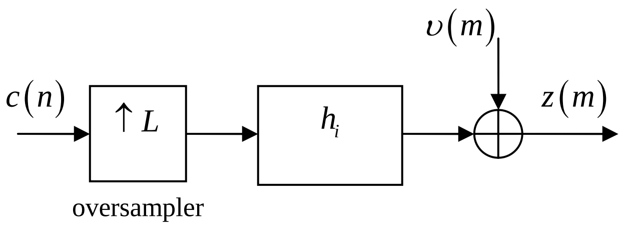

Figure 3.5: Magnetic recording channel.

The channel output $z(m)$ is given by (see Eq. (2.5), Chapter 2), for $L=1$,

$$z(m)=\sum_n c(n)h_{m-n}+\upsilon(m). \tag{3.3}$$

Then, we define the output SNR as

$$SNR_{out} \triangleq \frac{E\left[\left(\sum_n c(n)h_{m-n}\right)^2\right]}{E\left[\upsilon^2(m)\right]}, \tag{3.4}$$

where the noise power is given by $E\left[\upsilon^2(m)\right]=\frac{\sigma_u^2}{R}$. Ignoring correlation in the data $\{c(n)\}$, we get the power of the signal component as $E\left[\left(\sum_n c(n)h_{m-n}\right)^2\right]\approx\frac{E_b}{RT_u}$, where $E_b=\int h^2(t)\,dt$ is the energy of the bit response. In Appendix B, the energy of the bit response is proven to be

$$E_b\approx\frac{A^2T_u\log(3)}{3D_u}R^2, \tag{3.5}$$

which shows that E_b decreases quadratically with code rate. Therefore, the channel output SNR becomes

$$SNR_{out}=\frac{A^2\log(3)}{3\sigma_u^2D_u}R^2. \tag{3.6}$$

Because the output SNR varies in proportion to R^2, it is very clear that high rate codes are extremely important in recording systems. Otherwise, it is quite possible that the SNR loss due to the lower code rates may overwhelm the SNR gain provided by the code. Figure 3.6 illustrates the effect of code rate on the channel output SNR. In this plot, we have chosen the pulse amplitude to be $A = 1$, and σ_u^2 is chosen for $SNR = 30dB$ (see Eq. (2.7), Chapter 2). Figure 3.6 shows that the SNR at the channel output degrades significantly as the code rate decreases. For instance, at user density $D_u = 2.0$, the SNR is halved at the code rate $R = 0.75$ as compared to that at the code rate $R = 1$.

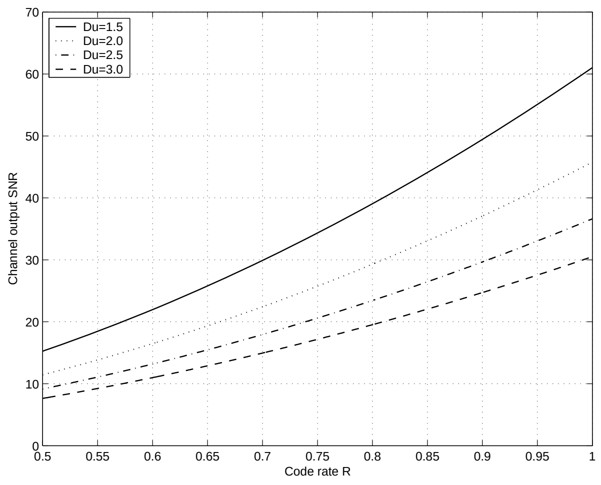

Figure 3.6: The effect of code rate on the channel output SNR for various densities.

3.2.2 Constrained Codes with Weak Constraints

In contrast to most distance-enhancing codes with limited code rate, high rate codes received special attention because they reduce the noise enhancement that results from the difficulty to equalize the read-back signal to the PR target at higher channel densities. The achievable code rate or capacity of the distance-enhancing codes discussed in

Section 3.1 are limited since these codes are expected to completely eliminate all the data patterns that support the identified dominant error events. In other words, these codes force the probability of data patterns that support the error event to be zero, *i.e.* $\pi(e) = 0$ in Eq. (2.22) (see Section 2.4.2). On the other hand, higher capacity codes can become possible if we insist only on partial elimination of the data patterns instead of complete elimination. For this type of codes, the effective minimum distance remains the same (see Eq. (2.23), Chapter 2). However, the probability $\pi(e)$ of the data patterns supporting the dominant error event is reduced. There are a number of types of high rate codes that serve different purposes, which are constructed based on this principle.

A class of MTR codes that do not completely eliminate the pairs of data patterns that support the dominant error event is called quasi-MTR or soft MTR codes. As an example, let us recall the identification of the modulo 2 time-varying MTR constraint in Section 3.1.2. To identify this constraint, the pairs of data patterns supporting the error string [0 0 2 -2 2 0 0] were required. We noted that all the pairs except two contained at least one pattern with 4 consecutive transitions. The constraint $MTR(j=3)$, which forbids 4 consecutive transitions in the input sequence, does not eliminate the dominant error event $\{2,-2,2\}$. Instead, the probability of the data patterns supporting the error event is decreased by a factor of 8, as only 2 pairs out of 16 contain data patterns which do not violate $MTR(j=3)$ constraint. This constraint has capacity 0.9468 and supports codes with rate as high as 16/17. Several rate 16/17 $MTR(j=3)$ codes have been proposed [22,30,18]. Weak constraints for forbidden list codes are identified in the same manner as weak constraints for MTR codes [68].

3.2.3 Survey of Encoder/Decoder Design Techniques

The construction of high-rate constrained codes is far from obvious, as table look-up approach for encoding and decoding such codes are impractical. Fortunately, a code with block encoder/decoder architecture can always be designed using the well known enumerative techniques [28]. The enumerative coding techniques make it possible to translate source words into codewords and vice versa by invoking an algorithmic procedure rather than performing the translation with a look-up table. Code rates very close to capacity can be achieved with the use of long codewords. However, severe error propagation results from the use of long codewords. This issue can be avoided by reversing the conventional order of the outer error correcting code and the inner constrained code [29,69].

Recently, Wijngaarden and Soljanin [30] presented advanced combinatorial techniques to design high-rate MTR block codes. With symmetry considerations, the set of source words and the set of candidate codewords are partitioned. The encoder/decoder mapping is then derived from the defined partitions. This methodology makes it possible to identify the best MTR and RLL constraints for a given code rate [30]. Furthermore, the constrained codes designed with that technique can serve as basic codes for the construction of even higher rate constrained codes [30].

3.3 Constrained Codes with Error Control Capabilities

In recording systems, until recently, there has been a strict separation between error control coding and modulation coding. In order to improve the overall efficiency, it has been proposed to combine parity-check code and modulation code [35,37,53]. Besides, for a given constrained code and error control requirement, it is usually not practical to design a specific error control code. In most combined schemes, parity bits are appended to the constrained codewords [31,32]. With the help of an error-control code, the parity bits are judiciously chosen such that the runlength constraints remain satisfied, as shown in Figure 3.7.

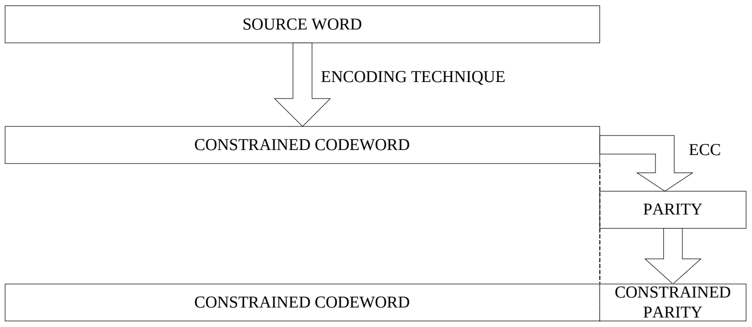

Figure 3.7: Conventional technique for inserting parity bits into a constrained sequence.

Wijngaarden and Immink [33] present special techniques for efficiently adding error control information without violating the imposed constraints. Coding algorithms based on enumeration techniques are applied to translate the user words into constrained codewords. The constrained codewords contain bits that can be flipped such that the resulting word still satisfies the runlength constraints. The positions of these bits are called ‘unconstrained positions’. Therefore, ECC parity bits can be inserted into the constrained words at the unconstrained positions. Wijngaarden and Immink [33] also present techniques to construct constrained codes which have a given number of unconstrained positions. The proposed techniques are based on enumerative coding algorithms or combinatorial circuitry. Campelo de Souza *et al.* [73] have presented design techniques for more general finite-state codes with unconstrained positions. Their work focuses on the design of MTR constrained finite-state codes. In particular, they propose construction techniques for MTR codes, with short block length, that improve on the techniques presented in [33]. Béal *et al.* [74] have presented time-efficient constructions of codes defined with a list of forbidden blocks and a set of unconstrained positions. These codes can be equipped with error control capabilities by simply reserving the unconstrained positions for the parity bits. The modulation encoder is followed by a precoder which performs NRZI to NRZ conversion. The methods developed in the above mentioned schemes [33, 73, 74] aim at adding error control capabilities to the constrained codewords before precoding. Cideciyan and Eleftheriou [34] have recently proposed a novel approach to combine modulation constraints with parity-check constraints at the output of the precoder. Because the motivation is the correction of short error events at the detector output, the authors identified the required set of parity-check equations at the precoder output, and translated them into a set of parity-check equations at the precoder input.

3.4 Conclusions

In this chapter, we reviewed techniques related to constrained modulation codes for PRML receivers. At the cost of efficiency, strongly constrained distance-enhancing codes completely eliminate the dominant error events at the output of the Viterbi detector. For improving the efficiency (*i.e.* code rate) of strongly constrained distance-enhancing codes, a strategy based on the identification of the error string to eliminate the error event is adopted. Distance-enhancing codes usually have low code rates. For this reason, they can be easily implemented via the state-splitting algorithm. The detection performance of PRML receiver is very sensitive to the code rate of the constrained code. This is shown in this chapter by examining the SNR at the output of the perpendicular recording channel, based on the ‘tanh’ model. Because of the importance of the code rate, methods based on relaxed runlength constraints allow the design of higher rate constrained codes. Techniques based on enumeration [28] or combinatorial circuitry [30] for designing encoder and decoder for high-rate weakly constrained codes require usually more effort, as compared to low-rate constrained codes. Implementing separately the ECC code (parity-check) and the constrained code may result in loss in efficiency. Whereas the conventional technique for equipping constrained code with error control capabilities is done by appending parity-check for each codeword, more efficient combination techniques based on the identification of unconstrained positions have been proposed recently in the literature.

Chapter 4

Parity-Check Codes and Post-Processing Techniques

As discussed in Chapter 3, the approach of using distance-enhancing constrained codes to achieve improved detection performance suffers from the problem of code rate loss. This arises because these codes use strong constraints to eliminate the dominant error events. The approach of using parity-check codes in combination with post-processing has been proposed [35] as an attractive ECC method for eliminating the dominant error events without suffering much rate loss. Moreover, as compared to powerful ECC schemes such as the turbo-codes which are known for their high complexity and latency [83], the approach of parity-check codes with post-processing offers an attractive trade-off between performance gain and implementation complexity. The basic idea is to use a parity-check code to detect the occurrence of an error event, from among a preselected list of dominant error events, at the detector output and to use a post-processor to locate and correct the errors. The basic principles of parity-based post-processing are explained in Section 4.1. Various parity-check codes are described in Section 4.2. In Section 4.3, we present two kinds of implementations of the post-processor, which are used widely. In this section, we also examine the optimality of these post-processors, and derive the optimum post-processor by formulating the problem of error event location as a multiple signals detection problem.

The original contributions in this chapter are as follows. Firstly, we derive the optimum MAP post-processor by considering the post-processor as a multiple signals detector. The analysis results in analytical expressions for the reliability information and, most importantly, for some normalization constants. Secondly, we show that the post-processor based on the computation of Euclidean distances is based on the ML criterion.

4.1 Principle of Parity-Based Post-Processing

4.1.1 Overview

Parity-based post-processing uses the error detection capability of a code in conjunction with the error location capability of a post-processor to detect and correct erroneous bits in the Viterbi detected sequence. Figure 4.1 is a block representation of the detector structure including the parity-based post-processor (PPP). The Viterbi detector (VD) does the preliminary detection of stored data bits by taking into account the runlength constraints of the constrained modulation code and the bit response of the partial response (PR) equalized recording channel. Here, Δ is the detection delay introduced by the Viterbi detector, and Δ' is the detection delay introduced by the overall detector. The PPP acts to correct the errors in the preliminary decisions. Details of how PPP does the detection and correction of errors are explained below.

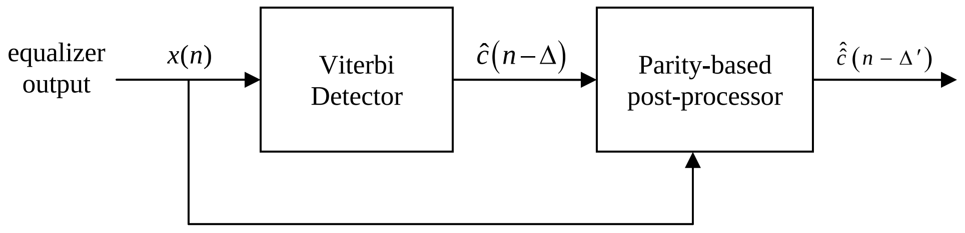

Figure 4.1: Structure of the overall detector for parity-coded channels.

The parity-based post-processor (PPP) has 3 main functions: parity check, error event detection and error correction, as shown in Figure 4.2. The PPP tries to locate and correct bit errors only if the parity constraints imposed by the parity-check code are violated. The parity constraint violation is highlighted by the 'parity check' block. In the 'error event detection' block, the post-processor attempts to identify the type as well as the location of the error event that has occurred. This error event search is limited to the set of dominant error events that satisfy the modulation constraints. For example, if the constrained code is a time-varying MTR code, then the error event type and location detected by the PPP must be such that the corrected sequence $\{\hat{\hat{c}}(n)\}$ of decisions must obey the time-varying MTR constraints. Obviously, the set of possible error events is also limited by the type of parity constraints violation in the parity codeword. The error event

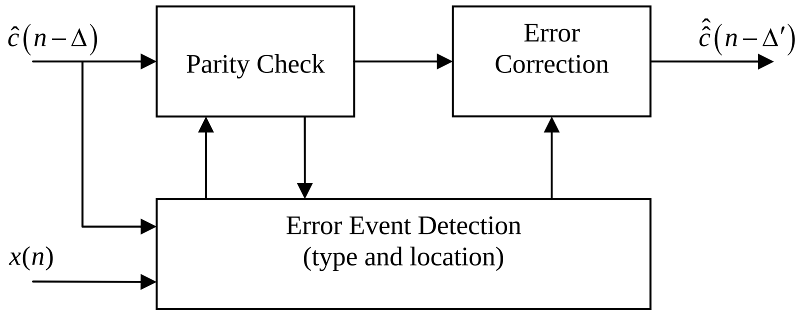
$\hat{c}(n-\Delta)$
Parity Check
Error Correction
$\hat{\hat{c}}(n-\Delta')$
Error Event Detection (type and location)
$x(n)$

Figure 4.2: Structure of the parity-based post-processor (PPP).

detection part is based on either Euclidean distances computation or matched filtering (see Figures 4.4 and 4.5). The details of the blocks in Figure 4.2 will be explained in Sections 4.2 and 4.3.

4.1.2 Post-processing algorithm

For clarification purpose, Figure 4.3 shows a flow chart that illustrates how the post-processor works on each codeword $\underline{\hat{c}}$ at the VD output for error detection and correction. There is parity violation if the syndrome of the detected codeword $\underline{\hat{c}}$ is different from 0. Section 4.2 explains how the syndrome is computed. If the syndrome is 0, the post-processor does not attempt to correct the current codeword $\underline{\hat{c}}$. On the other hand, a parity violation may lead to correcting the parity codeword.

The post-processor optimizes some reliability information terms $R_{i,j}$ with respect to the types of error events $i \in A$ and the location indices $j \in B$, where A is the set of all the valid error events, and B is the set of location indices in the codeword at which a valid error event may start. The set of valid error events and set of location indices are constrained by the runlength and parity constraints of the constrained parity-check code. Indeed, the reliability information $R_{i,j}$ for given i and j will be computed only if the codeword $\underline{\hat{\hat{c}}}$ that results from the correction of the detected codeword $\underline{\hat{c}}$ with the error event of type i starting at time j, satisfies both the runlength and parity constraints. For a given $\underline{\hat{c}}$, if a valid error event type and location are detected, then correction is performed. Otherwise, the erroneous parity codeword is left unchanged.

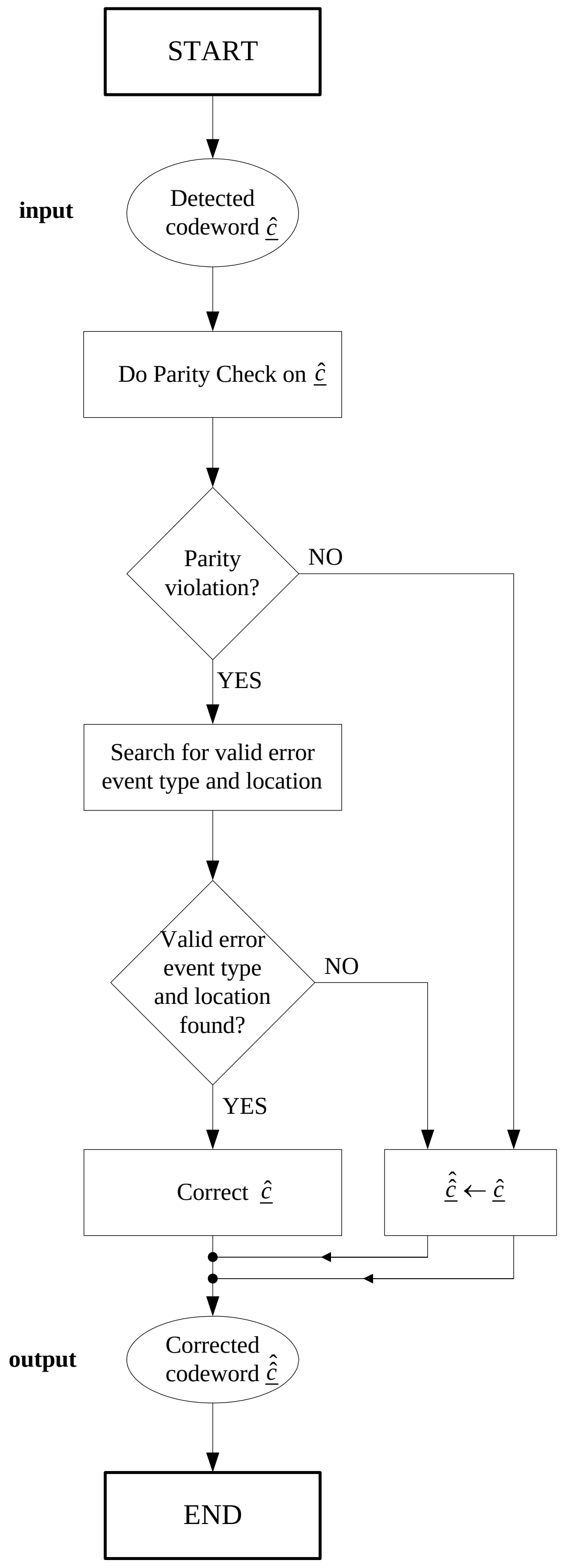
START
input
Detected codeword $\underline{\hat{c}}$
Do Parity Check on $\underline{\hat{c}}$
Parity violation?
NO
YES
Search for valid error event type and location
Valid error event type and location found?
NO
YES
Correct $\underline{\hat{c}}$
$\underline{\hat{\hat{c}}} \leftarrow \underline{\hat{c}}$
output
Corrected codeword $\underline{\hat{\hat{c}}}$
END

Figure 4.3: The parity-based post-processor algorithm.

4.2 Parity-Check (PC) Codes

A parity-check (PC) code is said to be a 'good' code if it results in minimum rate loss and maximum coding gain. In addition, the parity codeword must satisfy the modulation constraints used in the channel. Consequently, the problem of minimizing the rate loss while maintaining reasonable coding gain and the need to obey the modulation constraints are the major concerns in the design of parity-check codes for recording channels. Linear cyclic codes have been the most widely used PC codes [36] [37] [40] . Recently, methods have been proposed to insert PC information in constrained codes [33] [34] [38].

Linear cyclic codes represent a subset of the class of block codes which satisfy linear and cyclic properties [39]. The encoding operation performed by the PC code can be represented by the matrix equation[5]

$$\underline{b} = \underline{a}\underline{G}\,, \tag{4.1}$$

where $\underline{a} = \left[a(1),...,a(p)\right]$ is the source word (*e.g.* modulation constrained codeword), $\underline{b} = \left[b(1),...,b(q)\right]$ is the codeword and $\underline{G}$ is a $p \times q$ matrix, called the generator matrix. The code is cyclic if the codewords satisfy the cyclic shift property: if $\underline{b} = \left[b(1),...,b(q)\right]$ is a codeword, then the vectors obtained by cyclic shifts of the elements of $\underline{b}$ are also codewords. The key to the underlying structure of cyclic codes lies in the association of a polynomial with every source word and codeword.

The source word can be expressed as a $(p-1)^{th}$ order source polynomial

[5] In this section, addition and multiplication operations are defined modulo 2.

$$a(z) = a(1) + a(2)z + ... + a(p)z^{p-1}. \tag{4.2}$$

Similarly, the codewords can be expressed as $(q-1)^{th}$ order code polynomials. Then, there exists a unique polynomial $g(z)$ with minimal degree such that every code polynomial can be expressed as $b(z) = a(z)g(z)$ [39]. In practice, the generator polynomials are found by searching over all the irreducible factors of $z^q + 1$ in $GF(2)[z]$. Several interesting cyclic codes have been found, including the Hamming codes, Golay code, and BCH codes.

Let $\hat{b}(z) = b(z) + e(z)$ be the polynomial associated with the detected codeword, where $e(z)$ is the error polynomial. The decoder for linear cyclic codes computes the syndrome for the detected codeword. The syndrome polynomial, $s(z)$, is the remainder resulting from the division of $\hat{b}(z)$ by $g(z)$. If the syndrome is the zero-polynomial, then no error is detected. Otherwise, an error is detected, and the detected codeword is corrected with the most likely error pattern that makes the corrected codeword to have zero syndrome and satisfy the code constraints.

In PRML systems, the decoder for linear cyclic codes is simplified. Instead of considering a large number of error patterns, attention is focused on a few dominant error events. For a given order, the generator polynomial characterizes a linear cyclic code with given error detection capabilities. The higher the order of the generator polynomial is, the more error events which can be detected. But, the number of parity bits (*i.e.* $q - p$) is equal to the order of the generator polynomial. Therefore, use of more parity bits will increase the rate loss, unless the size of the source word (*i.e.* p) is also increased. Clearly, what is important is to find an optimum trade-off between rate loss and error detection capability. In practice, rate loss is minimized by using a few parity bits (*e.g.* 1 to 3) for the detection of a few dominant error events and long codewords [40] (*e.g.* 30-80 bits per parity bit).

Linear cyclic codes have proved to be very powerful. However, conventional linear cyclic codes do not introduce modulation constraints in the codewords. As discussed in Section 3.3, there is an increasing interest in combining the constrained modulation code and the PC code. The design techniques presented in Section 3.3 are extremely efficient. While the technique presented by Wijngaarden and Immink [33] takes advantage of the unconstrained positions in the constrained codewords, the constrained PC code presented by Cideciyan and Eleftheriou [34] inserts a few parity bits at selected positions to limit the degradation of the runlength constraints.

4.3 Parity-Based Post-Processing

We shall first describe the two implementations of parity-based post-processor (PPP) currently used in practice. Thereafter, we investigate the optimum post-processing approach and comment on the currently used implementations.

4.3.1 Current Implementations

Two implementations of PPP are currently used. One is based on the computation of Euclidean distances [40,32] and the other is based on a bank of matched filters [35,13,37].

The Euclidean distance based PPP scheme, shown in Figure 4.4, detects the error events such that the reconstructed signal based on the corrected codeword $\underline{\hat{\hat{c}}}$ is the “closest” to the received signal at the detector input. This amounts to computing the following reliability information values:

$$R_{i,j(Euc)} = \sum_n \left[x(n) - \left(\hat{c}(n) + e_i(n-j) \right) \otimes g_k \right]^2 \tag{4.3}$$

where g_k represents the PR target, $\hat{c}(n)$ is the detected sequence, $x(n)$ is the detector input, and $e_i(n-j)$ denotes the error event of type i starting at time j. The computation of $\min_{i,j} R_{i,j}$ gives the most probable type of error event and its location. The sets of valid error events and starting indices are determined by the syndrome result and modulation constraints.

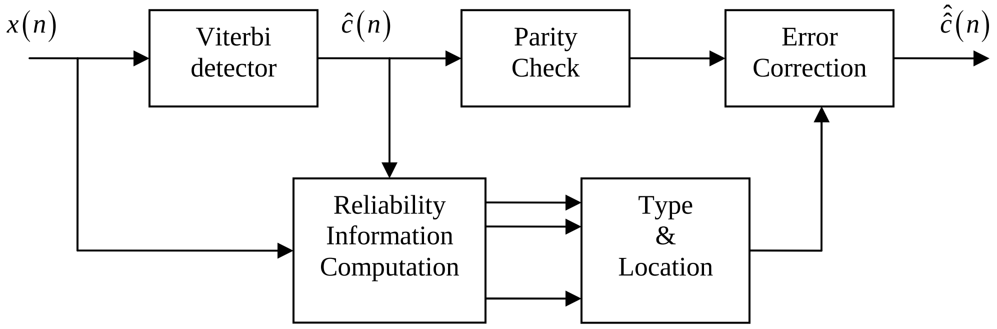

Figure 4.4: Post-processor based on Euclidean distances.

The second PPP scheme, shown in Figure 4.5, uses a set of matched filters for detecting the type and location of the error event. These matched filters are called error event matched filters since each filter is matched to the cascade of the PR target and one of the possible error events. In effect, the error event is detected by examining the error event matched filter (EEMF) that results in maximum SNR at its output compared to the rest. This amounts to computing the following reliability information values:

$$R_{i,j(MF)} = \left(x(n) - \hat{c}(n) \otimes g_k\right) \otimes \alpha_i(-n-j)\Big|_{n=0} + \theta_{i,j}, \tag{4.4}$$

where $\theta_{i,j}$ is a normalization constant, and $\alpha_i(n) = e_i(n) \otimes g_k$. The details of $\theta_{i,j}$ will be clarified in the next subsection.

The performance of both post-processors depends much on the number of parity bits and the design of the PC code. If an error event happens to occur on the boundary between two consecutive parity codewords, then the post-processor is likely to perform miscorrection[6]. Also, the usual assumption based on which these post-processors operate is that there is no more than one error event per detected codeword. Therefore, occurrence of multiple error events will also disturb the detection process.

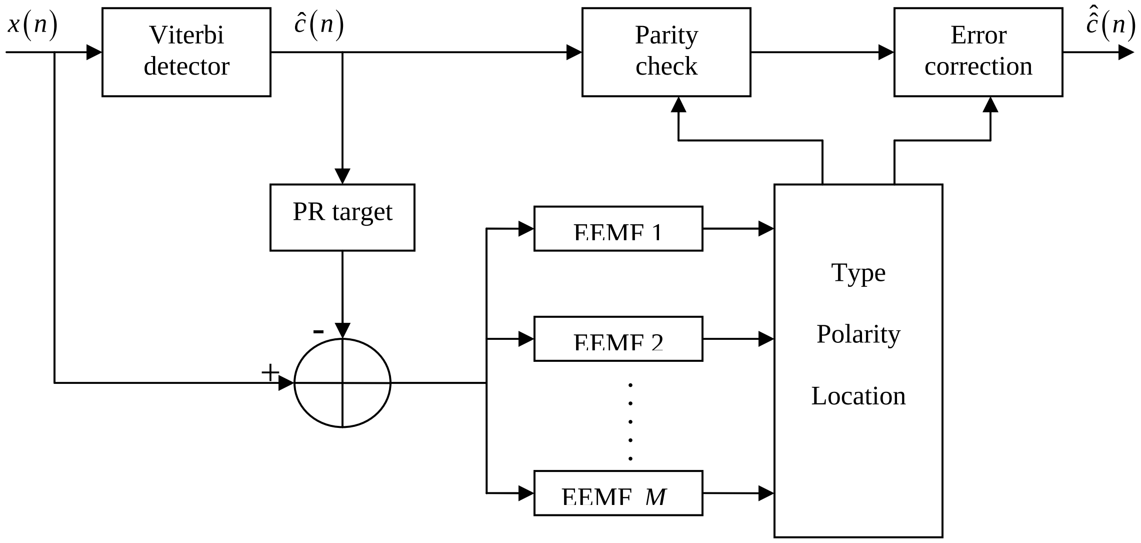

Figure 4.5: Post-processor based on error event matched filtering.

[6] This issue is tackled, in Chapter 5 and 6, by adjusting the post-processing algorithm presented in Subsection 4.1.2 to the joint detection of error event(s) in consecutive detected codewords $\underline{\hat{c}}$.

4.3.2 Optimum Post-Processor: Multiple Signals Detection Problem

The principle of the post-processor is to detect the error event that has occurred, from among a set of possible error events. As seen from Figure 4.1, the post-processor has two inputs: the detected sequence $\underline{\hat{c}}$ and the equalizer output $\underline{x}$. The detected sequence provides information on the syndrome of the received parity codeword. The equalizer output can be used to construct an error signal (see Eqns. (4.3) and (4.4)) as

$$e_x(n) = x(n) - \sum_{k=0}^{N_g - 1} g_k \hat{c}(n-k), \tag{4.5}$$

where N_g is the length of the PR target. Substituting for $x(n)$ from Eq. (2.12), we get (Note: $x(n) = \sum_k g_k c(n-k) + \eta(n)$)

$$e_x(n) = \sum_{k=0}^{N_g - 1} g_k e(n-k) + \eta(n), \tag{4.6}$$

where $e(n) = c(n) - \hat{c}(n)$ and $\eta(n)$ is the total noise at the detector input. Observe from Eq. (4.6) that the error signal can be considered as the output of a communication channel with channel bit response g_k and noise $\eta(n)$. This is illustrated in Figure 4.6.

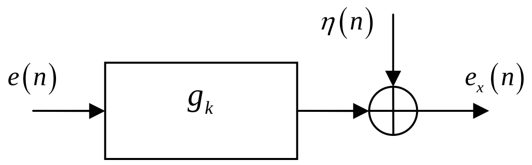

Figure 4.6: Communication channel formulation for the error signal $e_x(n)$ in Eq. (4.6).

Therefore, the task of the post-processor is to decide which error event is present in the signal $e_x(n)$, having been told that an error event has occurred. This can be done optimally using the maximum *a posteriori* (MAP) criterion.

Let $\underline{c} = \left[c(0),\ldots,c(N-1)\right]$ represent the transmitted codeword. The error sequence $\underline{\tilde{e}} = \underline{c} - \underline{\hat{c}} = \left[e(0),\ldots,e(N-1)\right]$ contains one of the possible error events $\underline{e}_{i,j}$ defined by

$$e_{i,j}(n) = \begin{cases} e_i(n-j) & \text{if} \quad j \le n \le j + L_{e_i} - 1 \\ 0 & \text{otherwise,} \end{cases} \tag{4.7}$$

where L_{e_i} is the length of the error event e_i. Thus, the MAP decision rule can be expressed as

$$\underline{\hat{e}}_{i_0,j_0} = \arg\max_{\underline{e}_{i,j}} \mathrm{p}\left(\underline{e}_{i,j} / \underline{e}_x\right), \tag{4.8}$$

where $\mathrm{p}\left(\underline{e}_{i,j} / \underline{e}_x\right)$ is the probability density function (pdf) of $\underline{e}_{i,j}$ conditioned on $\underline{e}_x$. Eq. (4.8) is equivalent to

$$\underline{\hat{e}}_{i_0,j_0} = \arg\max_{\underline{e}_{i,j}} I_{i,j}, \tag{4.9}$$

where $I_{i,j} \triangleq \log\left(\mathrm{p}\left(\underline{e}_x / \underline{e}_{i,j}\right)\right) + \log\left(\Pr\left[\underline{e}_{i,j}\right]\right)$ and $\Pr\left[\underline{e}_{i,j}\right]$ is the probability that the error event e_i occurs and starts at time index j. The error signal $\underline{e}_x$ in Eq. (4.6) is given by

$$e_x(n) = e_{x,i,j}(n) + \eta(n), \qquad 0 \le n \le N_c + N_g - 2, \tag{4.10}$$

where $e_{x,i,j}(n) = \sum_{m=0}^{N_g-1} g_m e_{i,j}(n-m)$. The noise $\eta(n)$ comprises the residual ISI and the equalized channel noise. Even if the residual ISI is assumed to be zero, the derivation of the optimum receiver leads to a complex detector [57]. Even though it is possible to use suboptimum schemes[7], we derive the optimum receiver for the case where the noise $\eta(n)$ is white Gaussian with variance σ_η^2. The pdf of $\eta(n)$ is given by

$$\mathrm{p}_\eta(x) = \frac{1}{\sqrt{2\pi}\sigma_\eta}\exp\left(-\frac{x^2}{2\sigma_\eta^2}\right). \tag{4.11}$$

Combining Eqns. (4.10) and (4.11), we get

$$\mathrm{p}\left(\underline{e}_x / \underline{e}_{i,j}\right) = \left(\sqrt{2\pi}\sigma_\eta\right)^{-(N+N_g-1)}\exp\left(-\sum_{n=1}^{N+N_g-1}\frac{\left(e_x(n)-e_{x,i,j}(n)\right)^2}{2\sigma_\eta^2}\right). \tag{4.12}$$

Therefore, the expression of $I_{i,j}$ can be simplified as

$$I_{i,j} = -\left(N+N_g-1\right)\log\left(\sqrt{2\pi}\sigma_\eta\right) - \sum_{n=0}^{N+N_g-2}\frac{\left(e_x(n)-e_{x,i,j}(n)\right)^2}{2\sigma_\eta^2} + \log\Pr\left[\underline{e}_{i,j}\right]. \tag{4.13}$$

Let us define $J_{i,j}$ as the part of $I_{i,j}$ which depends on i and j. Then, we get

$$\begin{aligned} J_{i,j} &\triangleq \sigma_\eta^2\left[I_{i,j} + \left(N+N_g-1\right)\log\left(\sqrt{2\pi}\sigma_\eta\right)\right] + \frac{1}{2}\sum_{n=0}^{N+N_g-2} e_x^2(n) \\ &= \sum_{n=0}^{N+N_g-2}\sum_{m=0}^{N+N_g-2} e_x(n)e_{x,i,j}(m) - \frac{1}{2}\sum_{n=0}^{N+N_g-2} e_{x,i,j}^2(n) + \sigma_\eta^2\log\Pr\left[\underline{e}_{i,j}\right]. \end{aligned} \tag{4.14}$$

Eq. (4.9) is therefore equivalent to

$$\hat{\underline{e}}_{i_0,j_0} = \arg\max_{\underline{e}_{i,j}}\left[\sum_{n=0}^{N+N_g-2}\sum_{m=0}^{N+N_g-2} e_x(n)e_{x,i,j}(m) + \theta_{i,j(MAP)}\right], \tag{4.15}$$

where $\theta_{i,j(MAP)} \triangleq -\frac{1}{2}\sum_{n=0}^{N+N_g-2} e_{x,i,j}^2(n) + \sigma_\eta^2\log\Pr\left[\underline{e}_{i,j}\right] = -\frac{1}{2}E_i + \sigma_\eta^2\log\Pr\left[\underline{e}_{i,j}\right]$, and E_i is the energy of the error signal $e_{x,i,j}(n)$. Thus, Eq. (4.15) can be rewritten as

[7] Suboptimum schemes can make the use of noise predictor filter to whiten the noise $\eta(n)$. However, by appropriately choosing the PR target, we can ensure that the noise $\eta(n)$ is close to white. We use the latter approach in Chapter 6.

$$\underline{\hat{e}}_{i_0,j_0} = \arg\max_{\underline{e}_{i,j}} \left[e_x(n) \otimes \alpha_i(-n-j)\Big|_{n=0} + \theta_{i,j(MAP)} \right], \tag{4.16}$$

where $\alpha_i(n) = e_i(n) \otimes g_k$. Eq. (4.16) shows that the optimum post-processor is formed with a bank of matched filters $\alpha_i(-n-j)$. The number of such filters is given by the number N_e of error events $\underline{e}_i$ to be detected. For each possible location j, the post-processor correlates the error signal $e_x(n)$ with the N_e matched filters $\alpha_i(-n-j)$.

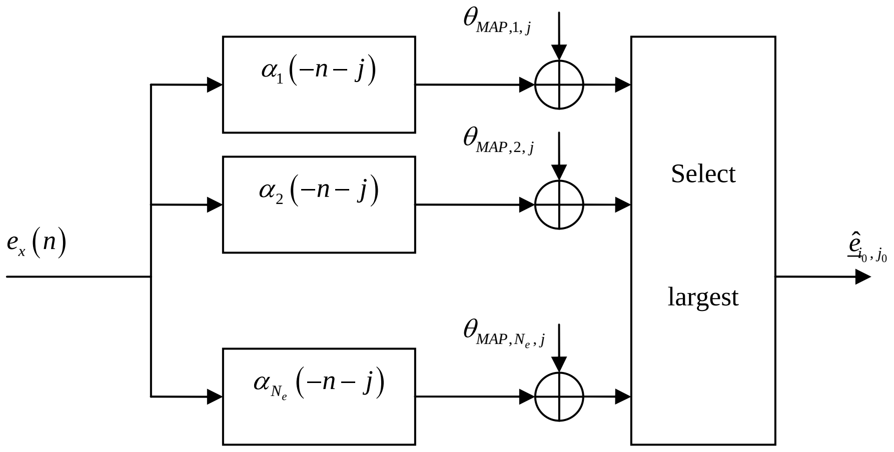

Figure 4.7: Optimum receiver structure of MAP post-processor.

Figure 4.7 shows the structure of the optimum MAP post-processor. Note that the post-processor based on matched filtering given in Figure 4.5 is the optimum MAP post-processor when the normalization constant $\theta_{i,j}$ in Eq. (4.4) is chosen to be $\theta_{i,j(MAP)}$ given in Eq. (4.15). We will analyze, in the next subsection, the optimality of the post-processor based on Euclidean distances (see Figure 4.4).

The normalization constant $\theta_{i,j(MAP)}$ depends on the probability $\Pr\left[\underline{e}_{i,j}\right]$, which is given by

$$\Pr\left[\underline{e}_{i,j}\right] = \Pr\left[\underline{c}, \hat{\underline{c}}\right]$$

where $\underline{\hat{c}}$ is the detected data at the Viterbi detector output (or post-processor input) and $\underline{c} = \underline{\hat{c}} + \underline{e}_{i,j}$ is the data sequence that supports the error sequence $\underline{e}_{i,j}$. Since the detected data $\underline{\hat{c}}$ is known to the post-processor, we may express $\Pr\left[\underline{e}_{i,j}\right]$ as

$$\Pr\left[\underline{e}_{i,j}\right] = \Pr\left[\underline{c}/\underline{\hat{c}}\right]\Pr\left[\underline{\hat{c}}\right], \tag{4.17}$$

where $\Pr\left[\underline{c}/\underline{\hat{c}}\right]$ is given in Eq. (C.14) (Appendix C), where $\underline{c}$ and $\underline{\hat{c}}$ are interchanged. In other words, $\Pr\left[\underline{c}/\underline{\hat{c}}\right]$ can be viewed as the probability of 'detecting $\underline{c}$ instead of $\underline{\hat{c}}$'. Consequently, we have

$$\Pr\left[\underline{c}/\underline{\hat{c}}\right] = Q\left(\frac{d_e^2 + 2m_Y}{2\sigma_{\tilde{Y}}}\right), \tag{4.18}$$

where m_Y and $\sigma_{\tilde{Y}}$ (see Eq. (C.14), Appendix C) represent the mean and the variance, respectively, of the Gaussian random variable $Y = \sum_n \sum_{k=0}^{N_g-1} g_k e_{i,j}\left(n-k\right)\eta\left(n\right)$. Thus, an accurate expression of the normalization constant $\theta_{i,j(MAP)}$ is given by

$$\theta_{i,j(MAP)} = -\frac{1}{2}E_i + \sigma_\eta^2 \log Q\left(\frac{d_e^2 + 2m_Y}{2\sigma_{\tilde{Y}}}\right) + \sigma_\eta^2 \log \Pr\left[\underline{\hat{c}}\right]. \tag{4.19}$$

Note that the third term $\sigma_\eta^2 \log \Pr\left[\underline{\hat{c}}\right]$ can be dropped, since it neither depends on i nor on j.

Instead of the MAP criterion, if we use the ML criterion for the detection of the error event, we will get

$$\underline{\hat{e}}_{i_0,j_0} = \arg\max_{\underline{e}_{i,j}} p\left(\underline{e}_x/\underline{e}_{i,j}\right) = \arg\max_{\underline{e}_{i,j}} I'_{i,j}, \tag{4.20}$$

where $I'_{i,j} \triangleq \log\left(\mathrm{p}\left(\underline{e}_x/\underline{e}_{i,j}\right)\right)$. Then, as in Eq. (4.15), we can show that

$$\underline{\hat{e}}_{i_0,j_0} = \arg\max_{\underline{e}_{i,j}} \left[\sum_{n=0}^{N+N_g-2} \sum_{m=0}^{N+N_g-2} e_x(n) e_{x,i,j}(m) + \theta_{i,j(ML)} \right], \tag{4.21}$$

where $\theta_{i,j(ML)} \triangleq -\frac{1}{2} E_i$.

4.3.3 Relationship between Post-Processors

In this subsection, we examine the relationship between post-processor PPP1 based on Euclidean distances and post-processor PPP2 based on error event matched filters, given in Figures 4.4 and 4.5, respectively. We just showed in Section 4.3.2 that the post-processor PPP2 is a MAP multiple signals detector when the normalization constants are selected according to Eq. (4.19). In order to examine the optimality of PPP1, we expand Eq. (4.3) to obtain

$$\begin{aligned} R_{i,j(Euc)} &= \sum_n x^2(n) + \sum_n \left(\hat{c}(n) \otimes g_k\right)^2 - 2\sum_n x(n)\left(c(n) \otimes g_k\right) + \sum_n \left(e_i(n-j) \otimes g_k\right)^2 \\ &\quad + 2\sum_n \left(\hat{c}(n) \otimes g_k\right)\left(e_i(n-j) \otimes g_k\right) - 2\sum_n x(n)\left(e_i(n-j) \otimes g_k\right) \\ &= \sum_n e_x^2(n) - 2\sum_n e_x(n)\alpha_i(n-j) + \sum_n \alpha_i^2(n-j). \end{aligned} \tag{4.22}$$

Since $\sum_n e_x^2(n)$ is the same for all i and j, this term can be dropped from Eq. (4.22). Further, noting that $E_i = \sum_n \alpha_i^2(n-j)$, we can express (4.22) as

$$R_{i,j(Euc)} = -\sum_n e_x(n)\alpha_i(n-j) - \theta_{i,j(ML)}. \tag{4.23}$$

Thus, PPP1 detects the error signal $\underline{\hat{e}}_{i_0,j_0}$ by maximizing $\sum_n e_x(n)\alpha_i(n-j)+\theta_{i,j(ML)}$.

Therefore, the post-processor based on Euclidean distances is a ML multiple signals detector. Furthermore, we note from Eq. (4.19) that

$$\theta_{i,j(MAP)}=\theta_{i,j(ML)}+\sigma_\eta^2\log Q\left(\frac{d_e^2+2m_Y}{2\sigma_{\tilde{Y}}}\right)+\sigma_\eta^2\log\Pr\left[\underline{\hat{c}}\right]. \tag{4.24}$$

4.4 Conclusions

In this chapter, the principles behind parity-based post-processors have been explained in detail. The post-processors use the error detection capability of a parity-check code to identify and locate error events at the Viterbi detector output and correct the detected sequence. Because of their strong error detection capability, linear cyclic codes are the most commonly used parity-check codes. However, these codes are not suitable when it comes to combining, for better efficiency, runlength constraints and parity-check constraints into one encoder. The performance of the overall detector is very sensitive to the code rate of the parity-check code. High-rate parity-check codes minimize the rate loss at the Viterbi detector. However, high-rate parity-check codes require to use very few parity bits, which limits the error event detection performance of the post-processor.

Post-processors are usually implemented with either computation of Euclidean distances or error event matched filtering. By formulating the error event location as a multiple signals detection problem, we have shown that the optimum post-processor, based on MAP criterion, constitutes of bank of matched filters whose outputs are precisely normalized. The normalization constant takes into account the data pattern that supports the type and the location of the error event to be detected. When the normalization constant of the post-processor based on error event matched filtering is chosen to be the optimum MAP normalization constant, the post-processor becomes optimum. On the other hand, the post-processor based on the computation of Euclidean distances is proven to be based on ML criterion.

Chapter 5

Novel Constrained Parity-Check Code

This chapter presents the design of a novel parity-check code combined with a strong distance-enhancing modulation code for perpendicular recording channels. For the sake of convenience, it is assumed in this chapter that the recording channel is perfectly equalized to the chosen PR target and that the noise at the detector input is white Gaussian. The design of the distance enhancing code is described in Section 5.1. The design and details of the new parity-check code are presented in Section 5.2. In Section 5.3, the performance of the novel constrained parity-check code in combination with a post-processor unit is examined. For the sake of completion, we also present some results on non-ideal equalized recording channels with colored noise.

The original contributions in this chapter are as follows. First, we have designed a novel constrained parity-check code and post-processor for improving the bit error rate performance on perpendicular recording channels. Second, we have derived an analytical upper-bound of the bit error rate for the block-coded ideal channel which will be described in Section 5.1. For the sake of convenience, the derivation is available in Appendix D.

5.1 Modulation Code Design

The design of distance-enhancing code follows the method outlined in Section 3.1.1.

5.1.1 Identification of Distance-Enhancing Runlength Constraint

First, error event characterization at the Viterbi detector output is performed for the ideal recording channel presented in Figure 5.1. Following the study given in Section 2.3.2, the PR target is selected as $\underline{g} = [1, 2, 3, 2, 1]^T$ to model the perpendicular recording channel (ideal). Further, as mentioned above, the noise at the detector input is assumed to be additive white Gaussian (AWGN) with variance σ_η^2 which is determined by the SNR defined as

$$SNR(dB) = 10\log_{10}\left(\frac{\|\underline{g}\|^2}{R\sigma_\eta^2}\right) \tag{5.1}$$

where $\|\underline{g}\|^2$ is the energy of the PR target $\underline{g}$ and R is the code rate.

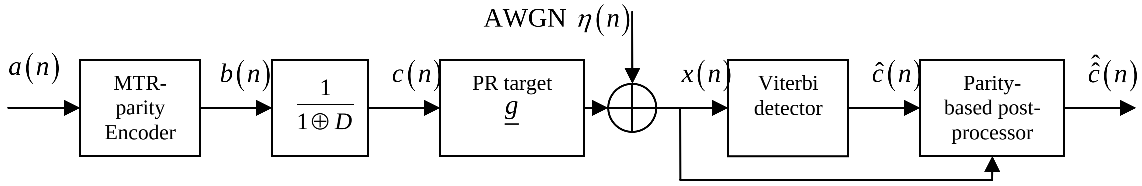

Figure 5.1: Schematic of a constrained parity-check coded ideal recording channel with Viterbi detector and parity-based post-processor.

Table 5.1 shows the distribution of the error events for the uncoded PR channel. This is based on a simulation collecting about 10000 error events. By examining the error events distribution, we note that 98% percent of the error events are of the type $\{+2, -2\}$,

$\{+2,-2,*,+2,-2\}$, $\{+2,-2,*,+2,-2,**,+2,-2\}$ … where * and ** denote strings formed by zero to $N_g - 1$ (with $N_g = 5$ here) consecutive 0's. Also shown in the table (in the last column) is the squared Euclidean distance $d_{\underline{e}}^2$ associated with each of the error events. Recall from Section 2.4.2 and Appendix C that these distances play a significant role in the BER performance of the Viterbi detector. In Section 5.1.3, we will use these distances for predicting the BER performance, coding gain, *etc*.

The dominant error events have a common structure, which is the presence of the error string $[0 \; +2 \; -2 \; 0]$ in almost all the error events. Therefore, removing this common

Table 5.1: Error events distribution for uncoded ideal PR channel.

No	Error Events											Number of occurences	Sq. Euclidean distance $d_{\underline{e}}^2$
1	+2	-2										8243	24
2	+2	-2	0	0	+2	-2						854	32
3	+2	-2	0	+2	-2							296	24
4	+2	-2	0	+2	-2	0	+2	-2				101	24
5	+2	-2	0	0	+2	-2	0	0	+2	-2		95	40
6	+2	-2	0	0	+2	-2	0	+2	-2			85	32
7	+2	-2	0	+2	-2	0	0	+2	-2			80	32
8	+2	-2	0	0	0	+2	-2					62	40
9	+2	-2	0	0	0	+2	-2	0	+2	-2		8	40
10	+2	-2	0	+2	-2	0	0	0	+2	-2		3	40
11	+2	-2	0	0	+2	-2	0	0	0	+2	-2	3	48
12	+2	-2	0	+2	-2	+2	-2					2	40
13	+2	-2	+2	-2	0	+2	-2					1	40
14	+2	-2	0	0	0	+2	-2	0	0	+2	-2	1	48
15	+2	-2	0	0	+2	-2	+2	-2				1	56
16	+2	-2	0	0	0	+2	-2	+2	-2			1	64

characteristic with a modulation code would eliminate a large number of error events. In fact, if we can prevent the error string $[0\ +2\ -2\ 0]$ from occurring, then all the error events listed in Table 5.1 can be eliminated. Before starting the analysis of the data patterns supporting this error string, we give in Figure 5.2 the schematic of the encoding scheme for the purpose of clarification.

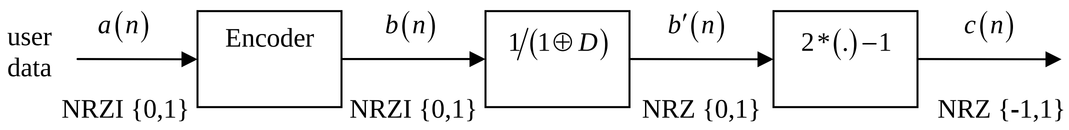

Figure 5.2: Schematic of the encoding shceme.

We now investigate the pairs of data patterns which support the error string $[0\ +2\ -2\ 0]$. These pairs are shown below in Figure 5.3 in terms of the NRZ data $\{c(n)\}$ as well as the NRZI data $\{b(n)\}$. By looking at the pairs of NRZI sequences

	0 2 -2 0	0 2 -2 0	0 2 -2 0	0 2 -2 0
$\underline{c}$	-1 1 -1 -1	-1 1 -1 1	1 1 -1 -1	1 1 -1 1
$\underline{\hat{c}}$	-1 -1 1 -1	-1 -1 1 1	1 -1 1 -1	1 -1 1 1
$\underline{b}$	1 1 0	1 1 1	0 1 0	0 1 1
$\underline{\hat{b}}$	0 1 1	0 1 0	1 1 1	1 1 0

Figure 5.3: Pairs of data patterns that support the error string $[0\ 2\ -2\ 0]$ in NRZ and NRZI formats.

$(\underline{\hat{b}},\underline{b})$, we can easily infer that the error string $[0\ +2\ -2\ 0]$ is eliminated if a code does not allow strings of two or more consecutive 1's in the NRZI sequence $\underline{b}$. This code is the $MTR(j=1)$ code that is equivalent to the $(d=1,k=\infty)$ RLL code. The maximum achievable code rate of this encoder is $\log_2\left(\left(1+\sqrt{5}\right)/2\right) \approx 0.6942$. With short word lengths, there is not much difficulty in designing such a code by invoking the state-splitting algorithm. Practical codes with rate 2/3 are available [84]. But, a code rate of 2/3 is too low for current magnetic recording systems. However, careful study of the data patterns (NRZI) in Figure 5.3 reveals that the error string $\left[0\ 2\ -2\ 0\right]$ can occur only if at least one of the following conditions is satisfied:

i) 2 consecutive transitions are allowed at even and odd time instants,

ii) 3 consecutive transitions are allowed at even or odd time instants.

Therefore, we can see that codes of higher rate may be obtained when time-varying constraints are considered. The MTR code with parameters $j_{odd} = 2$ and $j_{even} = 1$, denoted $MTR(1/2)$, eliminates the error string $\left[0\ 2\ -2\ 0\right]$. The constraint $j_{odd} = 2$ means that the encoder disallows 3 or more consecutive transitions starting at an odd time index, and $j_{even} = 1$ means that the encoder disallows 2 or more consecutive transitions starting at an even time index in the NRZI sequence $\underline{b}$. In other words, these constraints permit transition runs of only one and two at even and odd time indices, respectively. A transition diagram for the $MTR(1/2)$ constraints is given in Figure 5.4, where "square" and "circle" states correspond to even and odd time indices, respectively.

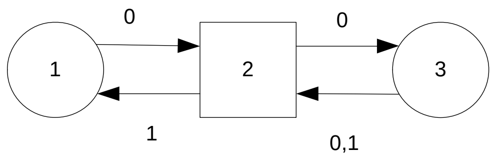

Figure 5.4: Graph representation of the $MTR(1/2)$ constraint.

Table 5.2 gives the encoding scheme of a rate 3/4 $MTR(1/2)$ self-concatenable block code obtained from [7,31]. This code also satisfies the RLL constraint $k=6$, which is why the code is also denoted as $MTR(1/2;6)$. Given by the logarithm of the largest eigenvalue of the transition matrix corresponding to the transition diagram, the capacity of the $MTR(1/2)$ constraint is 0.7925. Therefore, the rate 3/4 $MTR(1/2;6)$ block code given in Table 5.2 is 94.6% efficient.

Table 5.2: Look-up table for rate 3/4 MTR(1/2;6) code.

User bits $\{a(3n), a(3n+1), a(3n+2)\}$	Channel bits $\{b(4n), b(4n+1), b(4n+2), b(4n+3)\}$
000	0001
001	0010
010	0100
011	0101
100	0110
101	1000
110	1001
111	1010

5.1.2 Time-Varying Trellis

Because the runlength constraints in the $MTR(1/2;6)$ code are time-varying, the detector trellis also needs to be time-varying. In order to draw the time-varying trellis, we need to highlight the NRZ $\underline{c}$ patterns that are forbidden by the time-varying constraints. Because of the constraints $j_{even}=1$ and $j_{odd}=2$, the NRZI patterns $[b(2n-1), b(2n), b(2n+1)] = [0,1,1]$ and $[b(2n), b(2n+1), b(2n+2), b(2n+3)] = [0,1,1,1]$, respectively, are not allowed. These sequences correspond to $\left(c(2n-2), c(2n-1), c(2n), c(2n+1)\right) = (-1,-1,1,-1)$ or $(1,1,-1,1)$ for $j_{even} = 1$, and $\left(c(2n-1), c(2n), c(2n+1), c(2n+2), c(2n+3)\right) = (-1,-1,1,-1,1)$ or $(1,1,-1,1,-1)$ for $j_{odd} = 2$, in NRZ format. The detector trellis is designed to remove these 4 NRZ patterns. Since the

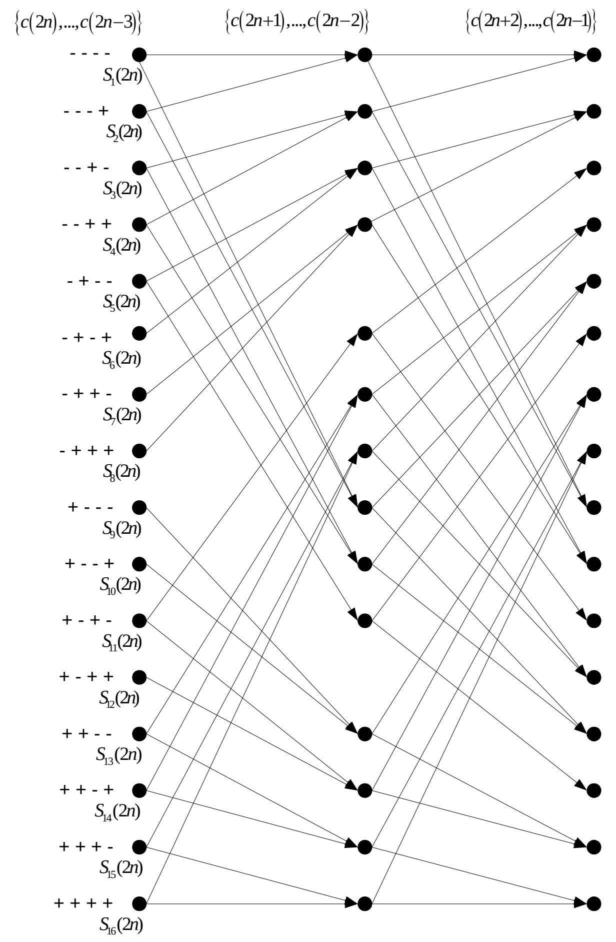

Figure 5.5: Structure of the time-varying detector trellis matched to the $MTR(1/2)$ constraints.

memory length of the PR target $\underline{g} = [1,2,3,2,1]$ is $N_g = 4$, the states in the trellis are defined as follows: $S_i(n) = \left(c(n), c(n-1), c(n-2), c(n-3)\right)$, where n is the time index and i is the state index. The first two forbidden sequences $\underline{c} = (-1,-1,1,-1)$ and

$\underline{c} = (1,1,-1,1)$ correspond to the states, $S_5(2n+1)$ and $S_{12}(2n+1)$, which can be eliminated by removing the incoming branches and the outgoing branches to and from these states. In the simulations, this is accomplished by assigning infinite metrics for these branches. The other two forbidden sequences, $\underline{c} = (-1,-1,1,-1,1)$ and $\underline{c} = (1,1,-1,1,-1)$, are of length 5 and these correspond to the state transitions $S_5(2n+2) \to S_{11}(2n+3)$ and $S_{12}(2n+2) \to S_6(2n+3)$, respectively. By removing these two branches, the detector is now matched to the MTR code. This means that no detected sequence can violate the MTR constraints. We do not account for the $k = 6$ constraint since the effect of this constraint on the detection performance is very weak (as k is large).

5.1.3 Coding Gain

To assess the gain in detection performance provided by the use of the $MTR(1/2;6)$ code, compared to the uncoded case, we now investigate the BER performance of the Viterbi detector. Noting that noise at the detector input is AWGN and the residual ISI is zero, recall from Section 2.4.2 and Appendix C that the BER at the Viterbi detector output can be evaluated as (*i.e.* a tight upper-bound)

$$P_b \simeq \sum_{\underline{e} \in \mathrm{E}} Q\left(\frac{d_e}{2\sigma_\eta}\right) w(e)\,\pi(e), \tag{5.2}$$

where $d_e = \left\| \underline{e} \otimes \underline{g} \right\|$ denotes the Euclidean distance associated with the error event $\underline{e}$, σ_η^2 is the variance of the AWGN, $\pi(e)$ is the probability of data patterns supporting the error event, $w(e)$ is the Hamming weight of the error event $\underline{e}$, and E is the set of all possible error events.

Table 5.1 shows that the BER performance of the uncoded PR channel is dominated by {+2,-2} error event. Further, with PR target $\underline{g} = [1,2,3,2,1]^T$, the Euclidean distance associated with the error event $\{+2,-2\}$ is $d_{\min}^2 = 24$. The detection SNR corresponding to this event is $\frac{24}{\sigma_\eta^2}$. The rate 3/4 $MTR(1/2;6)$ eliminates the $\{+2,-2\}$ error event. Table 5.3 shows the set of dominant error events and their Euclidean distances for the MTR coded channel. Observe that the $MTR(1/2;6)$ code has eliminated all the events in Table 5.1 containing the error string $[0\ 2\ -2\ 0]$. Further, the new dominant error event is $\{+2,0,-2\}$ with an associated squared Euclidean distance $d'^2_{\min} = 72$.The detection SNR corresponding to this event is $\frac{72}{\left(\sigma_\eta^2/R\right)}$ taking into account the code rate $R = 3/4$. Therefore, the resulting SNR gain (*i.e.* coding gain) is $20\log_{10}\left(\frac{\sqrt{R}d'_{\min}}{d_{\min}}\right) = 3.52\text{dB}$, which is quite significant. We will use this MTR code while designing a parity-check code to correct the new dominant error events at the detector output.

5.2 Design of Parity-Check Code and Post-Processor

The parity-check code is meant to facilitate detection and correction of the dominant error events that remain at the detector output. Because the *MTR*(1/2;6) block code is self concatenable, we choose to design a code having combined runlength and parity constraints by concatenating freely a certain number N_{MTR} of MTR codewords from Table 5.2 and appending a few parity bits. In Section 5.2.1, we present the structure of the MTR encoded sequence that defines the parity-check assignment. In Section 5.2.2, we show how to select the parity-check bits such that the $MTR(1/2)$ constraints are satisfied.

5.2.1 Data Structure for Parity Assignment

Our approach is based on the analysis of the data patterns supporting the dominant error events at the detector output. After removing all the error events containing the basic string $\{+2,-2\}$ (see Table 5.1) using the $MTR(1/2;6)$ code, almost all the remaining dominant error events are found to contain another basic string $\{+2,0,-2\}$ (see Table 5.3). Table 5.3 shows the dominant error events at the detector output for the $MTR(1/2;6)$ encoded PR channel.

Table 5.3: Squared Euclidean distances associated with the remaining dominant error events of the $MTR(1/2;6)$ coded channel.

Error event	Euclidean distance
$\underline{e}_1=\{+2,0,-2\}$	72
$\underline{e}_2=\{+2\}$	76
$\underline{e}_3=\{+2,0,-2,0,+2\}$	76
$\underline{e}_4=\{+2,0,-2,0,+2,0,-2\}$	80
$\underline{e}_5=\{+2,0,-2,0,+2,0,-2,0,+2\}$	84

For the construction of the parity-check code, we start by considering a single $MTR(1/2)$ codeword. Also, for the sake of clarity and convenience, we consider the pattern $\underline{b}' = \left(b'(n-1), b'(n), b'(n+1), b'(n+2), b'(n+3)\right)$ in NRZ{0,1} format instead of $\underline{c} = (c(n-1), c(n), c(n+1), c(n+2), c(n+3))$ in NRZ{-1,1} format (see Figure 5.2). Note that $c(n) = 2b'(n) - 1$. Table 5.4 presents the data patterns $\underline{b}'$ and $\underline{b} = (b(n), b(n+1), b(n+2), b(n+3))$ that support the error event $\{+2, 0, -2\}$. This listing shows that if n is even, then in six out of eight pairs of $\left(\hat{\underline{b}}, \underline{b}\right)$, one of the two patterns violates the MTR constraint $j_{even} = 1$. However, when n is odd, no pattern $\underline{b}$ violates the MTR(1/2) constraints. This means that the $MTR(1/2;6)$ code already reduces the probability of the error event $\{+2, 0, -2\}$. The design of the parity-check code is based on analysis of the data patterns which support this error event, as shown in Table 5.4.

Table 5.4: Data patterns, in NRZ{0,1} format, supporting the dominant error event $\{+2, 0, -2\}$.

	0 2 0 -2 0	0 2 0 -2 0	0 2 0 -2 0	0 2 0 -2 0
$\underline{b}'$	0 1 0 0 0	0 1 0 0 1	0 1 1 0 0	0 1 1 0 1
$\hat{\underline{b}}'$	0 0 0 1 0	0 0 0 1 1	0 0 1 1 0	0 0 1 1 1
$\underline{b}$	1 1 0 0	1 1 0 1	1 0 1 0	1 0 1 1
$\hat{\underline{b}}$	0 0 1 1	0 0 1 0	0 1 0 1	0 1 0 0
$\underline{b}'$	1 1 0 0 0	1 1 0 0 1	1 1 1 0 0	1 1 1 0 1
$\hat{\underline{b}}'$	1 0 0 1 0	1 0 0 1 1	1 0 1 1 0	1 0 1 1 1
$\underline{b}$	0 1 0 0	0 1 0 1	0 0 1 0	0 0 1 1
$\hat{\underline{b}}$	1 0 1 1	1 0 1 0	1 1 0 1	1 1 0 0

The single parity parity-check code, for which the parity bit is defined as

$$m = \left(\sum_{j=0}^{4N_{MTR}-1} b'(j) \right) \bmod 2, \tag{5.3}$$

with N_{MTR} denoting the number of concatenated $MTR(1/2;6)$ codewords, cannot detect the occurrence of $\{+2,0,-2\}$ error event. Indeed, the parity bit remains unchanged, because an even number of $b'(j)$ are flipped due to errors, as shown by Table 5.4. Therefore, we consider a dual-parity parity-check code which computes 2 parity bits for the even and odd subsequences. Unfortunately, such a parity-check code fails at detecting $\{+2,0,-2\}$ error event because the flipped bits, $b'(j)$ and $b'(j+2)$ are in the same subsequence. Therefore, we propose to divide the MTR encoded sequence $\{b'(n), n = 0,...,4N_{MTR}-1\}$ into M subsequences (with $M \geq 2$) in order to detect the occurrence of the dominant error event. Let us consider the set of indices $I = [0,...,4N_{MTR}-1]$. The indices corresponding to each subsequence are defined according to $I = \bigcup_{i=1}^{M} I_i$, with $I_i = \{2(Mk+i-1), 2(Mk+i-1)+1\}_{k=0,...,\frac{2N_{MTR}}{M}-1}$ for $i = 1,...,M$.

In other words, the set I_i defines the indices corresponding to the i^{th} subsequence. Each subsequence is assigned a parity obtained as

$$m'_i = \left(\sum_{j \in I_i} b'(j) \right) \bmod 2, \quad i = 1,...M. \tag{5.4}$$

Clearly, the parity bits $(m'_1,...,m'_M)$ associated with $\{b'(n), n = 0,...,4N_{MTR}-1\}$ are modified in two positions when the dominant error event $\{+2,0,-2\}$ occurs. Clearly, $M = 2$ will suffice to detect $\{+2,0,-2\}$. Therefore, the above described interleaving method and the definition of parity bit for each subsequence according to Eq. (5.4) can be used to successfully detect the error event $\{+2,0,-2\}$.

5.2.2 Selection of the Parity Bits

For the sake of convenience, we choose to append the parity bits to the MTR encoded sequence $\{b(n), n = 0,...,4N_{MTR}-1\}$ in NRZI format[8]. Note that the parity bits are designed based on the NRZ {0,1} data as given by Eq. (5.4). According to the subsequences defined in the previous section, at least 2 parity bits are required (*i.e.* $M \geq 2$) for the parity-check code to detect the error event $\{+2,0,-2\}$. We first investigate whether the parity bits (m_1', m_2') obtained from Eq. (5.4) can be freely appended to the MTR encoded

Table 5.5: Parity bits (m_1', m_2') associated with the $MTR(1/2;6)$ codewords.

$\{b(4n),..,b(4n+3)\}$	$\{b^{'}(4n-1),..,b^{'}(4n+3)\}$	$(m_1^{'}, m_2^{'})$
0 0 0 1	0 0 0 0 1	0 1
	1 1 1 1 0	
0 0 1 0	0 0 0 1 1	0 0
	1 1 1 0 0	
0 1 0 0	0 0 1 1 1	1 0
	1 1 0 0 0	
0 1 0 1	0 0 1 1 0	1 1
	1 1 0 0 1	
0 1 1 0	0 0 1 0 0	1 0
	1 1 0 1 1	
1 0 0 0	0 1 1 1 1	0 0
	1 0 0 0 0	
1 0 0 1	0 1 1 1 0	0 1
	1 0 0 0 1	
1 0 1 0	0 1 1 0 0	0 0
	1 0 0 1 1	

[8] In NRZI format, the set of $MTR(1/2;6)$ codewords makes it easier to design the parity bits. However, the design of parity bits in NRZ format may or may not be more efficient. We have not investigated this.

sequence $\{b(n), n=0,...,4p-1\}$. For this, we give in Table 5.5 the values of (m'_1, m'_2) when the number of concatenated codewords is $N_{MTR}=1$. Note that the parity bits m'_1 and m'_2 are computed using $\{b'(0), b'(1)\}$ and $\{b'(2), b'(3)\}$, respectively. The attempt to append (m'_1, m'_2) to $\{b'(2), b'(3)\}$ fails to satisfy the $MTR(1/2)$ constraints. Indeed, Table 5.5 shows that $\{b'(0), b'(1), b'(2), b'(3), m'_1, m'_2\}$ takes the value 010111 for the fourth $MTR(1/2)$ codeword. This sequence violates the $MTR(1/2)$ constraints since it contains a run of 3 consecutive '1'. In order to avoid this issue, we choose to design more parity-check bits. By choosing to append $(m'_1, 0, m'_2)$ to the MTR encoded sequence $\{b(n), n=0,...,4N_{MTR}-1\}$, the possibility of occurrence of 3 consecutive '1' can be avoided. However, such a design leads to a periodic inversion of the MTR constraints. This will require a more complex detector. With 4 parity bits, the design is straightforward. But, it is preferable (in view of rate loss) if we can manage with 3 parity bits instead of 4.

We consider $M=3$ subsequences instead of $M=2$. That is, we will have 3 parity bits. The $MTR(1/2;6)$ codewords help to select the parity bits. The resulting 3-tuple parity (m'_1, m'_2, m'_3) can be made to satisfy the $MTR(1/2)$ constraints by encoding these 3-tuples using the $MTR(1/2;6)$ encoding table given by Table 5.2. We append the resulting $MTR(1/2)$ complying encoded parity bits to the NRZI data as $[b(4N_{MTR}),...,b(4N_{MTR}+3)]$.

In the post-processor given in Section 4.1.1, the parity-check block checks for violation in the parity constraints. This is done as follows. Let

$\left[\hat{b}(4N_{MTR}),...,\hat{b}(4N_{MTR}+3)\right]$ represent the last 4 bits in NRZI{0,1} format resulting from the detected sequence $\left\{\hat{b}'(n)\right\}$. That is, $\hat{b}(4N_{MTR}+i)=\left(\hat{b}'(4N_{MTR}+i)+\hat{b}'(4N_{MTR}+i-1)\right)$ mod 2, for $i=0,...,3$. Then, the received version $(\hat{m}_1'',\hat{m}_2'',\hat{m}_3'')$ of (m_1',m_2',m_3') is obtained from Table 5.2 by applying the inverse mapping on channel bits $\left\{\hat{b}(4N_{MTR}+i),i=0,1,2,3\right\}$. Let $\hat{m}_i'$ be the parity of the i^{th} subsequence in $\left\{\hat{b}'(n)\right\}$, for $i=1,2,3$. Then, the syndrome is computed as $\hat{m}_i=(\hat{m}_i'+\hat{m}_i'')\text{mod } 2$, for $i=1,2,3$. Then, the parity-check block returns violation if the syndrome is $(\hat{m}_1,\hat{m}_2,\hat{m}_3)\neq 000$.

It is easy to see that the occurrence of any error event in Table 5.3 leads to parity violation. This parity-check encoder combining time-varying MTR constraints and parity constraints can detect all the 5 error events listed in Table 5.3. The rate of this newly formed constrained parity-check code (*i.e.* including MTR and parity-check encoding) is $R=3N_{MTR}/(4N_{MTR}+4)$. A reasonable value for N_{MTR} is 30. For large N_{MTR}, the error detection capability becomes degraded by the occurrence of multiple error events within a codeword. For small N_{MTR}, the rate loss becomes significant. Thus, the selection of the value of N_{MTR} defines the trade-off between rate loss and error detection capability.

5.2.3 Post-Processor Design

We shall now look at how to setup the post-processor to facilitate efficient and accurate detection and correction of the dominant error events for the $MTR(1/2;6)$ coded channel. We use the post-processors described in Section 4.2 for our study.

Let us first analyze how the dominant error events affect the parity bits. At moderate to high SNR, we can assume that at most one error event occurs in each parity codeword.

Table 5.6 gives the parity-check bits (*i.e.* syndrome bits) that result when the detected parity codeword contains any of the dominant error event listed in Table 5.3 starting at a time index which is in I_1, I_2 or I_3. If the parity-check bits are one of the 7 3-tuples $\hat{m}_1\hat{m}_2\hat{m}_3$ in Table 5.6, then the post-processor will search only among those error events which generate this 3-tuple. For example, if the parity-check is 010, then the post-processor tries to detect the error event $\{+2\}$ starting at a time index in I_2, or the error

Table 5.6: Parity-check (syndrome) $\hat{m}_1\hat{m}_2\hat{m}_3$ according to the starting index of the error event.

	Parity-check		
Error event	I_1	I_2	I_3
$\underline{e}_1=\{+2,0,-2\}$	110	011	101
$\underline{e}_2=\{+2\}$	100	010	001
$\underline{e}_3=\{+2,0,-2,0,+2\}$	111	111	111
$\underline{e}_4=\{+2,0,-2,0,+2,0,-2\}$	011	101	110
$\underline{e}_5=\{+2,0,-2,0,+2,0,-2,0,+2\}$	001	100	010

event $\{+2,0,-2,0,+2,0,-2,0,+2\}$ starting at a time index in I_3. The complexity of the post-processor is thus considerably reduced. The number of types of error events to be detected is always at most two and the number of location indices to be detected is less than $4N_{MTR}/3$. In terms of computational complexity, the proposed post-processor is approximately 6 times less complex compared to a post-processor that has to search through all the dominant events and locations.

Recall from Chapter 4 (Sections 4.2 and 4.3), we need to know the values of the normalization constants $\theta_{i,j}$ to implement the MAP-based error event matched filtering post-processor. The values of these normalization constants depend on the probabilities of the respective error events and associated data patterns (see Eq. (4.15)). The theoretical analysis required for deriving the expressions of these probabilities are given in Appendix D. In particular, note that the $MTR(1/2;6)$ coded data $\{c(n)\}$, $\{b'(n)\}$ or $\{b(n)\}$ is not wide-sense stationary. Instead, it turns out to be a cyclostationary process. This makes the theoretical analyses very involved and tedious. The details are given in Appendix D.

5.2.4 Coding Gain

We shall now evaluate the coding gain resulting from the use of the parity-check code designed above.

The parity-check code and post-processor is expected to detect and correct the five dominant error events listed in Table 5.3. This makes the new dominant error event at the post-processor output to be $\{+2,0,-2,0,+2,0,-2,0,+2,0,-2\}$ with squared Euclidean distance $d''^2_{\min}=88$. In our simulation studies, the number of MTR codewords used per parity-check codeword is $N_{MTR}=30$. This makes the code rate of the constrained parity-check code to be $R''=3N_{MTR}/(4N_{MTR}+4)=0.7258$. Since the squared minimum distance is $d'^2_{\min} = 72$ when only the MTR code with rate $R' = 0.75$ is used, we get the coding gain due to parity-check code alone as

$$20\log_{10}\left(\frac{\sqrt{R''}d''_{\min}}{\sqrt{R'}d'_{\min}}\right) = 0.729\text{dB}.$$

Thus, the total coding gain due to the constrained parity-check code is 3.52+0.729=4.25dB.

5.3 Simulation Results

In this section, we provide some simulation results to illustrate the effectiveness of the constrained parity-check code and post-processor developed in the previous sections. The user data $\{a(n)\}$ are chosen to be independent and identically distributed with $\Pr[a(n)=0] = \Pr[a(n)=1]$. The variance of the AWGN noise is chosen according to the SNR defined in Section 5.1.1. In our simulations, we do generate and transmit the parity bits explicitly unlike some of the earlier publications which assume that the correct parity bits are available to the receiver [85,86]. To tackle the issue of boundary error events, the parity-based post-processor algorithm given in Section 4.1.2 is slightly modified, by looking for boundary error events in case the syndromes of two consecutive parity codewords are both non zero.

Figure 5.6 shows the BER performance estimated at different points at the receiver under various conditions. The plot ‘uncoded’ means that neither the MTR code nor the parity-check code is used to encode the user data (*i.e.* code rate equal to 1), and the BER is estimated at the Viterbi detector output. This case serves as a reference to

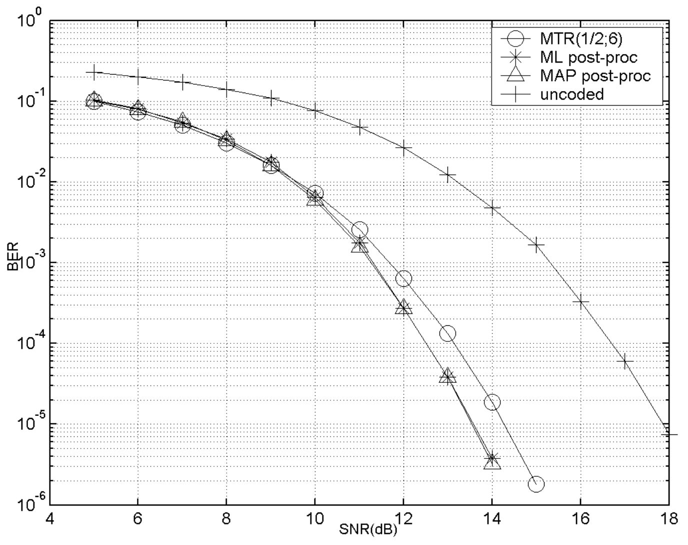

Figure 5.6: BER performance comparison with and without the constrained parity-check code.

measure the coding gains achieved from the channel codes used. The plot ‘MTR(1/2;6)’ implies that the channel code consists only of the rate 3/4 MTR(1/2;6) code, and the BER is estimated at the Viterbi detector output. As predicted in Section 5.1.3, the simulation results show a coding gain of 3.52dB with the MTR code compared to the uncoded case.

The plot ‘ML post-proc’ shows the BER performance obtained with the constrained parity-check code and ML post-processor (see Section 4.3 , Chapter 4). As expected, we obtain about 0.7dB gain in SNR with respect to the performance with MTR code only. Thus, we have a net coding gain of 4.2dB with the proposed constrained parity-check code.

Also shown in Figure 5.6 is the BER performance with the MAP post-processor (see plot ‘MAP post-proc’). Even though we expected the MAP to perform better than

ML, the results show that these post-processors result almost in identical performance. The reason for this can be understood by examining Table 5.7 which gives the probabilities of error events $P_{e_i,j} \triangleq \Pr\left[\underline{e}_{i,j}\right] = Q\left(\frac{d_{\underline{e}_i}}{2\sigma_\eta}\right)\pi_j\left(e_i\right)$ computed at SNR=12dB, where $\pi_j\left(e_i\right)$ is the probability of the data patterns supporting the error event $\underline{e}_i$ starting at a time index in the j^{th} interleave. Note from Eqns. (4.15) and (4.24) in Chapter 4 that this is the term that distinguishes between ML and MAP post-processors. Table 5.7 shows that these probabilities are comparable for the different error events. Consequently, the MAP and ML post-processor perform comparably.

Table 5.7: Probabilities $P_{e_i,j}$ of the dominant error events starting at a time index in the j^{th} interleave.

	$\underline{e}_1$	$\underline{e}_2$	$\underline{e}_3$	$\underline{e}_4$	$\underline{e}_5$
$j=0$	2.47e-5	5.33e-5	6.62e-6	1.58e-6	4.24e-7
$j=1$	2.47e-5	1.77e-5	1.77e-5	6.32e-6	4.52e-6
$j=2$	2.78e-5	5.33e-5	6.62e-6	1.78e-6	4.24e-7
$j=3$	2.47e-5	1.77e-5	8.83e-6	6.32e-6	2.26e-6

5.4 Performance Evaluation on Non-Ideal Channel

In the above sections, we modeled the perpendicular recording channel using an ideal PR target with $\underline{g} = [1,2,3,2,1]^T$, zero residual ISI, and AWGN noise at the detector input. But in practice, we use an equalizer to equalize the recording channel to the PR target $\underline{g}$. The resulting residual ISI will not be zero and the noise at the detector input will be a mixture of correlated Gaussian noise and residual ISI. In this section, we will do some performance evaluations to assess how the constrained parity-check code designed above will perform on such practical channels.

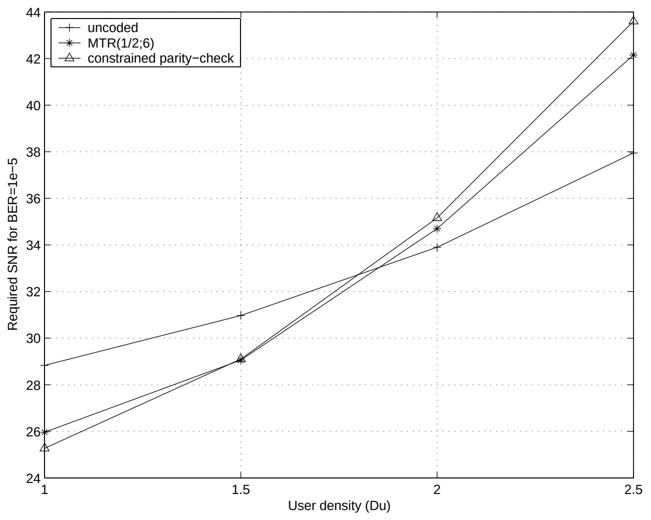

Figure 5.7: Required SNR for $BER = 10^{-5}$ versus user density for uncoded, $MTR(1/2;6)$ coded, and constrained parity-check coded channels.

In order to check if the $MTR(1/2;6)$ code brings about any improvement in the detection performance, the BER has been estimated for different user recording densities D_u using simulations. For the user densities $D_u = 1, 1.5, 2$ and $2.5,$ the SNR required to achieve a BER of 10^{-5} is plotted in Figure 5.7, considering the uncoded, $MTR(1/2;6)$ coded, and constrained parity-check coded channels. Several conclusions can be drawn from Figure 5.7. Firstly, Figure 5.7 shows that the BER performance of the $MTR(1/2;6)$ coded channel is better than that of the uncoded channel for densities $D_u \leq 1.8$. This means that the distance gain of the $MTR(1/2;6)$ code is larger than the rate loss for $D_u \leq 1.8$. Note that, at user density $D_u = 1.0$, the $MTR(1/2;6)$ code brings about 3dB SNR improvement as compared to the uncoded case. For larger densities $D_u \geq 1.8$, the rate loss overwhelms the distance gain of the $MTR(1/2;6)$ code. For instance, at $D_u = 2.0$, the $MTR(1/2;6)$ coded channel results in 0.8dB SNR loss as compared to the uncoded channel. Secondly, it can be seen from Figure 5.7 that the MAP-based post-processor results in better performance as compared to the $MTR(1/2;6)$ code, for user densities $D_u \leq 1.5$. For larger densities, the performance of the MAP-based post-processor is significantly degraded by rate loss and noise correlation. More precisely, even though the code rate of the constrained parity-check code (*i.e.* $R'' = 0.7258$) is close to that of the $MTR(1/2;6)$ code (*i.e.* $R' = 0.75$), the effect of code rate becomes more and more serious as density increases, as shown in Section 3.2.1 (Chapter 3). The optimality of the post-processors is based on the assumption that the noise at the Viterbi detector input is white Gaussian. However, in practice, the noise at Viterbi detector output is correlated and non-Gaussian (see Appendix C), and as density increases, the noise correlation becomes more serious. As a result, the post-processor is not optimal and its error event detection capability is reduced as density increases.

5.5 Conclusions

In this chapter, we have designed an original $MTR(1/2;6)$ constrained parity-check code for perpendicular recording channels equalized to $\underline{g} = [1,2,3,2,1]^T$ PR target. The design of the parity-check code is based on the self-concatenable time-varying $MTR(1/2;6)$ code [7, 31] which provides by itself a coding gain of 3.52dB as compared to the uncoded channel. Parity information is assigned to a certain number (*e.g.* 30) of concatenated $MTR(1/2;6)$ codewords by judiciously dividing the MTR encoded sequence and appending 4 parity bits. The performance of the constrained parity-check code combined with the post-processor described in Chapter 4 has been evaluated analytically and by using simulations. The proposed constrained parity-check code and post-processor provide 4.2dB SNR gain over the ideal uncoded PR channel. For a non-ideal channel, the proposed scheme shows a performance gain only for low densities. This is mainly due to the rate loss resulting from the constrained parity-check code.

Chapter 6

High-rate Constrained Codes and Post-Processing

As mentioned in Chapter 3, the performance of distance-enhancing constrained codes is significantly degraded by rate loss. Therefore, high-rate constrained codes receive particular interest. It is proposed in this Chapter to focus on a particular class of high-rate constrained codes, known as forbidden list (FL) constrained codes [54]. Generally speaking, the constraint strings, which constitute the forbidden list, eliminate specific data patterns which support a given error string. Since the set of data patterns supporting a given error string can be large, there is usually a wide range of choice in selecting high-capacity FL constraints. Such flexibility makes FL constraints very attractive. This flexibility will be used to match the FL constraints to the MAP post-processor. The identification of FL constraints may require a search that can be extremely complex. In Section 6.1, we propose a method for reducing the search range of the FL constraints. The design of FL constrained codes that have error control capabilities is not considered in this Chapter. Instead, we shall generate the constrained sequences from a suitably chosen information source, as explained in Section 6.2. In Section 6.3, we select specific parity-check codes and we simulate the performance of MAP and ML-based post-processors, in a data-aided mode.

The original contributions in this thesis are as follows. First, we present a method for designing very high-capacity FL constraints targeted at error events which are matched to MAP-based post-processors, *i.e.* wider separation between the probabilities of the different dominant error events. Second, we propose a method for building an information source that translates the designed FL constraints. Third, we show how to build the Viterbi trellis matched to the FL constraints without much increase in computational complexity.

6.1 Forbidden List Constraints

In this section, we present the methodology adopted for identifying high-capacity FL constraints. Based on an error event characterization for the PRML receiver studied in Chapter 2, we identify the set of eligible strings that will form the FL constraints. Thereafter, we propose a method for reducing the complexity of the search of FL constraints.

6.1.1 Design Methodology

In order to enhance the performance of PRML receivers, constrained codes aim at reducing the probability of dominant error events P_e (see Eq. (2.17)). FL constrained codes can be viewed as a general class of distance-enhancing codes (*i.e.* weakly or strongly constrained). While strong distance-enhancing constrained codes enhance the minimum effective detection SNR, SNR_{eff} (see Eq. (2.21)), by completely eliminating all the data patterns that support the dominant error events, weakly constrained codes do not enhance SNR_{eff} . The effective distance $d_{eff}\left(e\right)$ (see Eq. (2.23)) of the dominant error events is not increased. Instead, the probability $\pi\left(e\right)$ of the data patterns supporting the dominant error events is reduced. The design of high-rate FL constrained codes follows similar steps as the design of strong distance-enhancing codes presented in Section 3.1. More precisely, the design of high-rate FL constrained codes involves the following steps:

(a) Error event characterization for the given channel and identification of the dominant error events $\left\{\underline{e}_1,...,\underline{e}_N\right\}$.

(b) Determination of a list of error strings $\mathcal{E}=\left\{\underline{\varepsilon}_1,...,\underline{\varepsilon}_N\right\}$ that will prevent the occurrence of the error events in Step (a).

(c) Identification of a list of forbidden data strings $\mathcal{L}=\left\{\underline{s}_1,...,\underline{s}_M\right\}$ of given length that maximize the number of data patterns that support the error strings in Step (b), and tuning the Viterbi detector (VD) to the identified FL constraint.

(d) Construction of an efficient code.

Step (a) is the same as the first step of the design of strong distance-enhancing codes (see Section 3.1). In Step (b), the error strings $\underline{\varepsilon}_i$ are obtained by prefixing and postfixing the error events $\underline{e}_i$ with a sufficiently large number of zeros. This is crucial for not excessively restraining the set of eligible FL constraints in Step (c). In general, FL constraints are time-varying. As mentioned in Chapter 3, time-varying constraints can offer better performance as compared to time-invariant constraints, because of their higher capacity. Immink [78] shows that runlength constraints expressed in NRZ format are in general more attractive than NRZI runlength constraints. However, for a certain class of codes, Chaichanavong and Marcus [71] show that the choice between NRZ and NRZI formats depends on the case at hand. We choose to express the constraints in NRZI format. Therefore, the list of forbidden data strings will be denoted $\mathcal{L} = \{\underline{s}_1, ..., \underline{s}_M\}_{NRZI}$. For Step (c), computer search is performed to identify the eligible list of forbidden strings which minimize the BER and separate the probabilities of the two first dominant error events. The search may be prohibitively complex. For this reason, one should judiciously reduce the search range. A simple method for tuning the VD to the identified FL constraint is to choose the number of trellis states as $2^{L_{\max}-1}$, where $L_{\max}$ is the length of the longest string $\underline{s}_j$, $j = 1, ..., M$, and eliminate the branches that contain any of the data strings $\underline{s}_j$, $j = 1, ..., M$. If $L_{\max}$ is much greater than the memory of the partial response (PR) target, this method is extremely demanding in terms of computational complexity. We will show in Section 6.1.2 how this can be avoided with the help of a special state-tagging. For Step (d), high-rate code design has been reviewed in Section 3.2.3 (Chapter 3). Most importantly for the work done in this thesis, Béal [74] shows efficient algorithms to help the design of time-varying FL constrained codes that have error control capabilities. However, we limit the work in this thesis to time-invariant FL constraints. Because of time limitation, we do not design an explicit encoder. Instead, we shall generate the FL constrained sequences with a suitably chosen information source, as it will be explained in Section 6.2.

6.1.2 Identification of FL Constraints

The system which we simulate is shown in Figure 2.6. As mentioned in Sections 2.3 and 4.3, monic-constrained generalized partial response (GPR) target are interesting for two reasons. First, they result in near optimal performance, in the sense of minimizing the dominant error event probability. Secondly, they result in close-to-white noise at the VD input, which helps to enhance the performance of MAP and ML-based post-processors. For illustration purpose, we plot in Figure 6.1 the power spectral density of the noise at the VD input, when the recording channel is equalized to the $\underline{g} = [1, 2, 3, 2, 1]^T$ PR target and to the monic constrained GPR target of length 5. The SNR (see Eq. (2.7)) is chosen to be 33dB and the user density 2.0. Clearly, the noise correlation is much less when the monic-constrained GPR

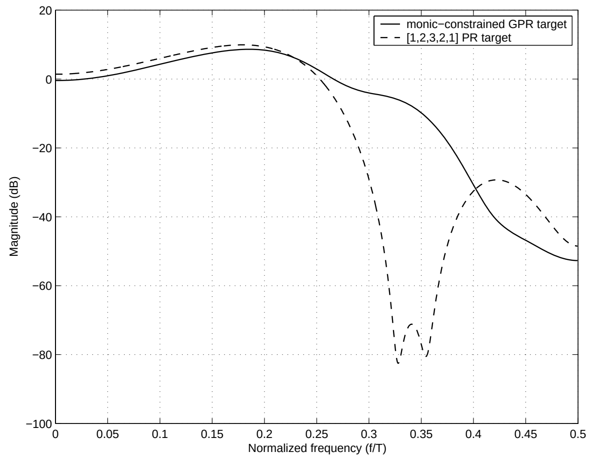

Figure 6.1: Normalized power spectral density of equalized noise at VD input for $[1, 2, 3, 2, 1]^T$ PR target and the monic-constrained GPR target of length 5.

target is used. Because of the reduction in noise correlation and the BER performance enhancement, we choose to use GPR targets in this chapter. We also choose to fix the user density to $D_u = 2.0$ throughout this chapter. In order to identify the dominant error events at the VD output, error event characterization is performed by analytically computing the probability P_e of any possible error event. Since long error events are less likely to occur, we restrain our search to error events whose length is less than 11. The SNR chosen is such that the BER is in the order of 10^{-5}. We choose $SNR = 33\text{dB}$. Table 6.1 lists the dominant error events and their probabilities of occurrence. Table 6.1 also displays the effective distance and the probability of the data patterns supporting the dominant error events.

Table 6.1: Dominant error events at the Viterbi detector output for uncoded GPR equalized channel.

Dominant error events $\underline{e}_i$	Effective distance $d_{eff}(e_i)$	Probability $\pi(e_i)$ of data patterns supporting $\underline{e}_i$	Probability P_{e_i} $Q\left(d_{eff}^2(e_i)\right)\pi(e_i)$
$\underline{e}_1 = \{+2,-2\}$	3.001	½	5.47e-6
$\underline{e}_2 = \{+2,-2,0,0,+2,-2\}$	3.319	1/8	1.68e-7
$\underline{e}_3 = \{+2,-2,0,+2,-2\}$	3.474	1/8	5.59e-8
$\underline{e}_4 = \{+2,-2,0,0,0,+2,-2\}$	3.463	1/8	1.52e-8
$\underline{e}_5 = \{+2,-2,0,+2,-2,0,+2,-2\}$	3.666	1/32	1.35e-8
$\underline{e}_6 = \{+2,-2,0,+2,-2,0,0,+2,-2\}$	3.586	1/32	6.18e-9
$\underline{e}_7 = \{+2,-2,0,0,+2,-2,0,+2,-2\}$	3.586	1/32	6.18e-9
$\underline{e}_8 = \{+2,-2,0,0,+2,-2,0,0,+2,-2\}$	3.616	1/32	4.95e-9
$\underline{e}_9 = \{+2,-2,0,+2,-2,0,+2,-2,0,+2,-2\}$	3.505	1/128	2.80e-9
$\underline{e}_{10} = \{+2,-2,0,0,+2,-2,0,0,0,+2,-2\}$	3.945	1/32	3.78e-10
$\underline{e}_{11} = \{+2,-2,0,0,0,+2,-2,0,0,+2,-2\}$	3.944	1/32	3.80e-10
$\underline{e}_{12} = \{+2,-2,0,+2,-2,0,0,0,+2,-2\}$	3.976	1/32	2.93e-10
$\underline{e}_{13} = \{+2,-2,0,0,0,+2,-2,0,+2,-2\}$	3.977	1/32	2.92e-10

Table 6.1 shows that the dominant error events are $\underline{e}_1 = \{+2, -2\}$, $\underline{e}_2 = \{+2, -2, 0, 0, +2, -2\}$, and $\underline{e}_3 = \{+2, -2, 0, +2, -2\}$. We notice that the error events have a common structure formed with the basic string $[2, -2]$. Because of this special feature, the set of forbidden data strings corresponding to the different dominant error events will show similarities. As a result, it will be possible to reduce the search range of the forbidden data strings. Nevertheless, this feature does not favor the search of FL constraints which increase the separation between the probabilities of the dominant error events. The maximal length of the forbidden strings $\underline{s}_j$ to be identified is limited by the maximal length of the error strings $\underline{\varepsilon}_i$. The main advantage is the flexibility that longer strings offer. More precisely, eliminating a given set of data patterns may result in higher capacity FL constraints when longer FL strings $\underline{s}_j$ are considered. Obviously, this flexibility can be exploited at the cost of larger complexity, *i.e.* with a larger number of FL strings $\underline{s}_j$. An error string corresponding to the first dominant error event is $\underline{\varepsilon}_1 = [0\ 0\ 0\ +2\ -2\ 0\ 0\ 0]$ of length $L_{\varepsilon_1} = 8$. There are $2^{L_{\varepsilon_1} - w(\varepsilon_1)} = 64$ different pairs $(\underline{c}, \underline{\hat{c}})$ of data patterns that support the error string $\underline{\varepsilon}_1$, where $\underline{c}$ and $\underline{\hat{c}}$ are data patterns in NRZ {-1,1} of the same length as $\underline{\varepsilon}_1$, that satisfy $\underline{c} - \underline{\hat{c}} = \underline{\varepsilon}_1$, and $w(\varepsilon_1)$ represents the Hamming weight of the error string $\underline{\varepsilon}_1$. The pairs of data patterns $(\underline{c}, \underline{\hat{c}})$ are translated into pairs $(\underline{b}, \underline{\hat{b}})$ in NRZI {0,1} format. For clarity, Table 6.2 displays 8 of the 64 pairs of data patterns $(\underline{b}, \underline{\hat{b}})$ that support the error string $\underline{\varepsilon}_1$. For the purpose of illustration, Table 6.2 highlights the string $[1\ 1\ 0\ 1]$ that eliminates one data pattern in 4 of the 8 pairs of data patterns. It can be shown this string is contained in 43 out of the 64 pairs $\left(\underline{b},\widehat{\underline{b}}\right)$. Equivalently, only 21 pairs out of 64 do not contain the string $\left[1\,1\,0\,1\right]$. Therefore, if we forbid the string $\left[1\,1\,0\,1\right]$, the probability of the data patterns that support the error string $\underline{\varepsilon}_1$ is reduced by a factor of 64/21.

Table 6.2: 8 pairs of data patterns in NRZI format which support the error string $\left[0\ 0\ 0\ +2\ -2\ 0\ 0\ 0\right]$.

0	0	1	1	0	0	0
0	0	0	1	1	0	0
0	0	1	1	0	0	1
0	0	0	**1**	**1**	**0**	**1**
0	0	**1**	**1**	**0**	**1**	1
0	0	0	1	1	1	1
0	0	**1**	**1**	**0**	**1**	0
0	0	0	1	1	1	0
0	0	1	1	1	1	0
0	0	0	1	0	1	0
0	0	1	1	1	1	1
0	0	0	1	0	1	1
0	0	1	**1**	**1**	**0**	**1**
0	0	0	1	0	0	1
0	0	1	1	1	0	0
0	0	0	1	0	0	0

Table 6.3 lists the forbidden strings $\mathcal{L}^{(1)} = \left\{\underline{s}_1^{(1)},\ldots,\underline{s}_{M_1}^{(1)}\right\}$ for the error string $\underline{\varepsilon}_1$. The strings $\underline{s}_j^{(1)}$ are selected in the same manner as for the string $\left[1\,1\,0\,1\right]$ in Table 6.2. Obviously, longer strings are contained in fewer pairs $\left(\underline{b},\widehat{\underline{b}}\right)$, as compared to shorter strings. As we will see in Section 6.2, short strings result in FL constraints that have capacities satisfying $Cap < 0.9$. Such capacities are considered low capacities. Therefore, we select the strings whose length is larger than 4. However, it is possible that some strings of length 4 have low capacities. For instance, the FL constraint $\mathcal{L} = \left\{1110\right\}_{NRZI}$ has

capacity 0.8791. For verification of this result, one may apply one of the methods described in Section 6.2. Let $S^{(1)} = \left\{ \underline{s}_j^{(1)} \right\}_{j=1,\ldots,N^{(1)}}$ represent the set of eligible strings of length between 4 and 8. Table 6.3 shows that the subset of $S^{(1)}$ comprising the strings $\underline{s}_j^{(1)}$ of length 4 and 5 is relatively large. With a similar procedure applied to error strings

Table 6.3: List of eligible strings of length 4 and 5 which partially eliminate the pairs of data patterns supporting $[0\ 0\ 0\ +2\ -2\ 0\ 0\ 0]$ error string.

String $\underline{s}_j^{(1)}$	Ratio of elimination of pairs $(\underline{b},\hat{\underline{b}})$	Capacity	String $\underline{s}_j^{(1)}$	Ratio of elimination of pairs $(\underline{b},\hat{\underline{b}})$	Capacity
$[1\,1\,1\,0]$	50/64	0.8791	$[1\,0\,1\,1\,0]$	23/64	0.9543
$[0\,1\,1\,1]$	45/64	0.8791	$[1\,1\,0\,1\,0]$	23/64	0.9468
$[1\,0\,1\,1]$	44/64	0.9005	$[0\,1\,1\,1\,0]$	22/64	0.9510
$[1\,1\,0\,1]$	43/64	0.9005	$[1\,1\,1\,0\,1]$	22/64	0.9510
$[1\,1\,1\,1]$	38/64	**0.9468**	$[1\,0\,1\,1\,1]$	21/64	0.9510
$[1\,0\,0\,1]$	34/64	0.9005	$[1\,1\,0\,1\,1]$	21/64	0.9573
$[1\,0\,1\,0]$	34/64	0.9132	$[1\,1\,1\,1\,1]$	20/64	**0.9752**
$[1\,1\,0\,0]$	32/64	0.8791	$[0\,0\,1\,1\,1]$	16/64	0.9468
$[0\,1\,0\,1]$	31/64	0.9132	$[0\,1\,0\,1\,1]$	16/64	0.9468
$[0\,1\,1\,0]$	30/64	0.9005	$[0\,1\,1\,0\,1]$	16/64	0.9543
$[0\,0\,1\,1]$	29/64	0.8791	$[1\,0\,1\,0\,0]$	16/64	0.9468
$[0\,1\,0\,0]$	24/64	0.9005	$[1\,1\,0\,0\,1]$	16/64	0.9510
$[0\,0\,0\,1]$	22/64	0.8791	$[1\,0\,0\,1\,0]$	15/64	0.9543
$[0\,0\,1\,0]$	22/64	0.9005	$[1\,0\,0\,1\,1]$	15/64	0.9510
$[1\,0\,0\,0]$	16/64	0.8791	$[0\,1\,0\,1\,0]$	14/64	**0.9619**
$[0\,0\,0\,0]$	8/64	**0.9468**	$[1\,0\,1\,0\,1]$	14/64	**0.9619**
$[1\,1\,1\,1\,0]$	31/64	0.9468	$[0\,0\,1\,1\,0]$	12/64	0.9510
$[0\,1\,1\,1\,1]$	24/64	0.9468	$[0\,1\,1\,0\,0]$	12/64	0.9510
$[1\,1\,1\,0\,0]$	24/64	0.9468	$[1\,1\,0\,0\,0]$	12/64	0.9468

$\underline{\varepsilon}_2 = [0\ 0\ 2\ -2\ 0\ 0\ 2\ -2\ 0\ 0]$ and $\underline{\varepsilon}_3 = [0\ 0\ 2\ -2\ 0\ 2\ -2\ 0\ 0]$, we identify the sets $S^{(2)} = \left\{\underline{s}_j^{(2)}\right\}_{j=1,\ldots,N^{(2)}}$ and $S^{(3)} = \left\{\underline{s}_j^{(3)}\right\}_{j=1,\ldots,N^{(3)}}$. The set of all possible forbidden data strings associated with the error strings $\underline{\varepsilon}_1$, $\underline{\varepsilon}_2$ and $\underline{\varepsilon}_3$ is given by

$$S \triangleq \bigcup_{i=1}^{3} S^{(i)} . \tag{6.1}$$

The FL constraints are formed by taking any possible n-tuple of different string $\underline{s}_j$ in S. More precisely, the set of FL constraints is given by

$$\mathbb{L} \triangleq \left\{ \left\{\underline{s}_j\right\}_{j=1,\ldots,n} \Big/ \underline{s}_j \in S, n = 1,\ldots,|S| \right\} . \tag{6.2}$$

The total number of FL constraints $\mathcal{L} \in \mathbb{L}$ is $N_L = |\mathbb{L}| = 2^{|S|} - 1$, where $|\mathbb{L}|$ and $|S|$ denote the cardinalities of the sets $\mathbb{L}$ and S, respectively. Unfortunately, the number N_L is extremely large. For instance, when S contains only 30 strings, the BER and the error event probabilities need to be evaluated about 10^9 times. Fortunately, some simplifications are possible.

6.1.3 Reduction of the Search Range

The identification of FL constraints in Step (c) of the code design procedure given in Section 6.1.1 is limited by the prohibitively large number of possibilities. Since the main objective of FL constraints is to minimize the BER, we assess the strength of the different FL constraints by evaluating the BER at the VD output, for the system shown in Figure 6.2. The constrained source generates input data sequences that satisfy the given FL constraint. Details of the design of the constrained source are given in Section 6.2. The precoder translates the data sequence $\{b(n)\}$ in NRZI{0,1} format to the data sequence $\{c(n)\}$ in NRZ{-1,1} format. Since the number of different FL constraints is very large, estimating the BER using simulations would require phenomenal computation time. For this reason, we estimate the BER analytically. The details of the computation are given in Appendix C and Section 6.3.

Simulations show that[9] the BER becomes large when the input data is constrained, as compared to the uncoded case. This is expected since the FL constraints do not enhance the effective distance. For minimizing the rate loss, we consider

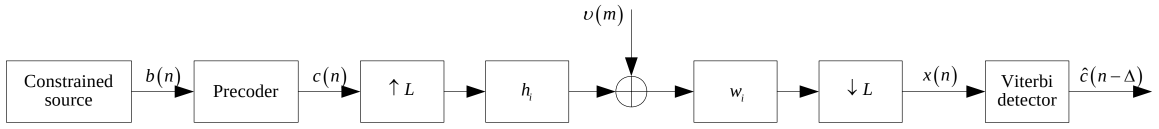

Figure 6.2: Schematic of a constrained perpendicular recording channel with Viterbi detector.

FL constraints $\mathcal{L}$ with capacity $Cap(\mathcal{L}) \geq 0.97$. For lower capacity constraints, the post-processor will not be able to provide satisfactory gain to compensate for the rate loss. We do the search for the 'optimal' FL constraint in two steps. In the first step, we reduce the search area by focusing on FL constraints that minimize the BER. In the second step, we identify in the reduced search area the FL constraints that separate the dominant error event probabilities. Obviously, the FL constraints which minimize the BER are constituted of strings with rather large lengths. The resulting BER performance becomes close to that of the uncoded channel. For avoiding this issue, we shall consider FL constraints $\mathcal{L}$ that contain strings of equal length L_s. With smaller length L_s (*e.g.* 4 or 5),

[9] These simulation results are not shown here.

the capacity of the resulting constraints $\mathcal{L}$ will be less than 0.97. Note that, for a given length L_s, the capacity of a FL constraint $Cap(\mathcal{L})$ decreases as the number of strings $\underline{s}_j \in \mathcal{L}$ increases. This is the reason why we choose $L_s = 8$. With larger lengths L_s, the number of constraints becomes impractically large.

For clarity, we summarize in detail the identification of FL constraints in Step (c) of the code design given in Section 6.1.1:

(c.1) Choose a fixed length L_s for the strings $\underline{s}_j \in \mathcal{L}$.

(c.2) For the error strings $\underline{\varepsilon}_i$ which correspond to the three most dominant error events $\underline{e}_i$, identify the strings $\underline{s}_j$ of length L_s which reduce significantly the number of supporting data patterns.

(c.3) Estimate the BER associated with each string $\mathcal{L} = \{\underline{s}_j\}$. Let $\tilde{S}$ denote the set of 15 strings $\underline{s}_j$ which result in smallest BER.

(c.4) Form all possible FL constraints $\mathcal{L} = \{\underline{s}_1, ..., \underline{s}_k\}$, where $k \le 5$ and $\underline{s}_j \in \tilde{S}$.

(c.5) Estimate the BER associated with each FL constraint and compute the ratio between the probabilities of the two most dominant error events.

(c.6) Select the FL constraints which maximize the probability ratio computed in Step (c.5) and result in acceptable BER.

For Step (c.1), the length L_s is chosen to be 8. In Step (c.2), note that the error strings $\underline{\varepsilon}_i$ chosen in Step (b) should be long enough to support the search of strings $\underline{s}_j$ of length L_s. Special consideration must be taken into account when identifying the strings $\underline{s}_j$. It is possible that two FL constraints $\mathcal{L}_1 = \{\underline{s}_{j_1}\}$ and $\mathcal{L}_2 = \{\underline{s}_{j_1}\}$, where $\underline{s}_{j_1}$ and $\underline{s}_{j_2}$ have different lengths, constrain the data sequence in exactly the same way. For instance, $\mathcal{L}_1 = \{[1\,1\,1\,1\,1\,1\,1\,0]\}$ and $\mathcal{L}_2 = \{[1\,1\,1\,1\,1\,1\,1]\}$ are equivalent FL constraints. This situation arises because the Shannon cover [75] of $\mathcal{L}_1$ has more than one irreducible component [75], whereas the Shannon cover of $\mathcal{L}_2$ is irreducible. More details about irreducibility are available in Appendix A and [75]. Therefore, strings $\underline{s}_j$ of length L_s which are equivalent to strings of length $L_s - 1$ or less are not selected. In Step (c.3), we select only the 12 strings $\underline{s}_j$ which result in smallest BER. Considering more than 12 will result in prohibitively large number of combination of strings for Step (c.4). In Step (c.4), we reduce the search range by considering only those FL constraints that contain at most 5 strings. The reason for this is that we consider only FL constraints $\mathcal{L}$ with $Cap(\mathcal{L}) \geq 0.97$, and in practice, FL constraints with more than 6 strings will not satisfy this inequality. In Step (c.5), the error event probabilities and the BER are estimated analytically. In Step (c.6), we choose the FL which maximize the separation between the probabilities P_{e_1}, P_{e_2} and P_{e_3} of the dominant error events $\underline{e}_1$, $\underline{e}_2$ and $\underline{e}_3$, respectively.

Table 6.4 lists the strings $\underline{s}_j \in \tilde{S}$ which are obtained in Step (c.3) of the above presented identification procedure. There is no specific rule for selecting the constraints that maximize the separation between error event probabilities. For this reason, we select the FL constraints which separately maximize the ratios $P_{e_{i_1}} / P_{e_{i_2}}$ and $P_{e_{i_2}} / P_{e_{i_3}}$, where $\underline{e}_{i_1}$, $\underline{e}_{i_2}$ and $\underline{e}_{i_3}$ are the ordered dominant error events. The simulations performed for Step (c.5) show that the order of dominance of error events $\underline{e}_1$, $\underline{e}_2$ and $\underline{e}_3$ remains unchanged

Table 6.4: Set of strings $\underline{s}_j$ of length 8 whose associated FL constraint $\mathcal{L} = \{\underline{s}_j\}$ result in smallest BER.

Constraint $\mathcal{L} = \{\underline{s}_j\}$	Capacity $Cap(\mathcal{L})$	BER	Constraint $\mathcal{L} = \{\underline{s}_j\}$	Capacity $Cap(\mathcal{L})$	BER
$\underline{s}_1 = [1\,1\,1\,1\,1\,1\,1\,1]$	0.9971	1.282e-5	$\underline{s}_7 = [1\,0\,0\,1\,1\,1\,1\,1]$	0.9942	1.428e-5
$\underline{s}_2 = [1\,1\,0\,1\,1\,0\,1\,1]$	0.9950	1.400e-5	$\underline{s}_8 = [1\,1\,1\,1\,1\,0\,0\,1]$	0.9942	1.428e-5
$\underline{s}_3 = [1\,1\,1\,0\,0\,1\,1\,1]$	0.9945	1.402e-5	$\underline{s}_9 = [1\,1\,0\,1\,1\,1\,0\,1]$	0.9946	1.435e-5
$\underline{s}_4 = [1\,1\,0\,0\,1\,1\,1\,1]$	0.9943	1.414e-5	$\underline{s}_{10} = [1\,0\,1\,1\,1\,0\,1\,1]$	0.9946	1.435e-5
$\underline{s}_5 = [1\,1\,1\,1\,0\,0\,1\,1]$	0.9943	1.414e-5	$\underline{s}_{11} = [1\,1\,1\,1\,1\,1\,0\,0]$	0.9942	1.438e-5
$\underline{s}_6 = [1\,0\,1\,1\,0\,1\,1\,0]$	0.9949	1.415e-5	$\underline{s}_{12} = [0\,0\,1\,1\,1\,1\,1\,1]$	0.9942	1.438e-5

for almost all the FL constraints. The main reason for this is that most strings $\underline{s}_j$ which eliminate some data patterns which support an error string $\underline{\varepsilon}_{i_1}$ also contribute to eliminate data patterns which support another error string $\underline{\varepsilon}_{i_2}$. Unfortunately, this property restrains the search for FL constraints which maximize the probability ratios mentioned above. Table 6.5 shows the FL constraints obtained which have maximum ratios $P_{e_{i_1}} / P_{e_{i_2}}$ or $P_{e_{i_2}} / P_{e_{i_3}}$. Also shown in Table 6.5 is the capacity and the BER associated with the respective FL constraint, whose strings $\underline{s}_j$ are listed in Table 6.4.

Table 6.5: FL constraints which result in maximum separation between the probabilities of dominant error events.

FL constraint $\mathcal{L}$	Capacity $Cap(\mathcal{L})$	BER	$P_{e_{i_1}} / P_{e_{i_2}}$	$P_{e_{i_2}} / P_{e_{i_3}}$
$\mathcal{L}_1 = \{\underline{s}_1, \underline{s}_5, \underline{s}_7, \underline{s}_{11}\}$	0.9847	1.91e-5	46.32	1.92
$\mathcal{L}_2 = \{\underline{s}_2, \underline{s}_6, \underline{s}_9, \underline{s}_{10}\}$	0.9856	1.95e-5	25.09	5.80
uncoded	$\times$	1.19e-5	32.56	3.00

As compared to the uncoded case, the ratios $P_{e_{i_1}} / P_{e_{i_2}}$ and $P_{e_{i_2}} / P_{e_{i_3}}$ obtained for the FL constraints in Table 6.5 have not varied much. The similarities between the set of forbidden strings for the different dominant error event can account for this result. When strings are chosen to be slightly shorter (*i.e.* $L_s = 7$), the separation between the probabilities is better. However, this is done at the cost of rate loss, as shown in Tables 6.6 and 6.7.

Table 6.6: Set of strings $\underline{s}_j$ of length 7 whose associated FL constraint $\mathcal{L} = \{\underline{s}_j\}$ result in smallest BER.

Constraint $\mathcal{L} = \{\underline{s}_j\}$	Capacity $Cap(\mathcal{L})$	BER	Constraint $\mathcal{L} = \{\underline{s}_j\}$	Capacity $Cap(\mathcal{L})$	BER
$\underline{s}_{13} = [1\,1\,1\,1\,1\,1\,1]$	0.9941	1.392e-5	$\underline{s}_{19} = [1\,1\,1\,1\,0\,0\,1]$	0.9883	1.722e-5
$\underline{s}_{14} = [1\,1\,0\,1\,1\,0\,1]$	0.9897	1.665e-5	$\underline{s}_{20} = [0\,1\,1\,1\,0\,1\,1]$	0.9889	1.758e-5
$\underline{s}_{15} = [1\,0\,1\,1\,0\,1\,1]$	0.9897	1.665e-5	$\underline{s}_{21} = [1\,1\,1\,1\,1\,0\,0]$	0.9881	1.759e-5
$\underline{s}_{16} = [1\,1\,0\,0\,1\,1\,1]$	0.9887	1.711e-5	$\underline{s}_{22} = [0\,0\,1\,1\,1\,1\,1]$	0.9881	1.759e-5
$\underline{s}_{17} = [1\,1\,1\,0\,0\,1\,1]$	0.9887	1.711e-5	$\underline{s}_{23} = [1\,0\,1\,1\,1\,1\,1]$	0.9883	1.797e-5
$\underline{s}_{18} = [1\,0\,0\,1\,1\,1\,1]$	0.9883	1.722e-5	$\underline{s}_{24} = [1\,1\,1\,1\,1\,0\,1]$	0.9883	1.797e-5

Table 6.7: FL constraints which result in maximum separation between the probabilities of dominant error events.

FL constraint $\mathcal{L}$	Capacity $Cap(\mathcal{L})$	BER	$P_{e_{i_1}} / P_{e_{i_2}}$	$P_{e_{i_2}} / P_{e_{i_3}}$
$\mathcal{L}_3 = \{\underline{s}_{13}, \underline{s}_{16}, \underline{s}_{17}, \underline{s}_{22}\}$	0.97063	2.91e-5	67.22	1.20
$\mathcal{L}_4 = \{\underline{s}_{14}, \underline{s}_{15}, \underline{s}_{20}\}$	0.9764	2.60e-5	20.29	9.85

6.2 Information Source

The design of high-rate constrained codes is found to be complicated. Instead of encoding the user bit sequences into channel bit sequences that satisfy the desired runlength constraints, we choose to generate the channel data via a suitably built information source. A highly efficient code will have similar statistics. We present a method for constructing such a source for a given list of forbidden constraints. First, we determine a deterministic transition diagram for the given FL constraints. Thereafter, we detail the characteristics of the information source.

6.2.1 Transition Diagram

FL constraints are conveniently represented with finite-state transition diagrams. We shall build such a representation in two steps. First, we build a trellis type transition diagram to represent the given FL constraint. Based on the finite-state form of this trellis diagram, an algorithm, called the 'Moore algorithm' [75], is applied to minimize the number of states of the finite-state diagram.

Let $\mathcal{L} = \left\{ \underline{s}_j, j = 1, \ldots, M \right\}_{NRZI}$ be the set of forbidden strings. Let $\underline{s}_{j_0}$ be a string with maximal length $L_{\max}$. Then, we define the $2^{L_{\max}-1}$ states[10] $\left(b\left(n - L_{\max} + 2\right), \ldots, b\left(n\right)\right)$ where the bits $b\left(n\right)$ represent the constrained coded binary input data to the channel in NRZI format. From the fully unconstrained trellis, we remove the branches that have labels equal to the forbidden strings with length $L_{\max}$. Then, we eliminate the states whose labels contain the strings $\underline{s}_i$ with length strictly less than $L_{\max}$. Thus, the unconstrained trellis diagram is transformed into a constrained trellis diagram which is matched to the FL constraint $\mathcal{L}$. The finite-state transition diagram is induced from the constrained trellis. The finite-state diagram may be reducible, in which case the diagram is reduced to one of its irreducible components [75]. It is important to note that the

[10] Note that the trellis states are defined in reverse time order as compared to the definition given in Section 2.4 (Chapter 2). This definition results in a property which is convenient for identifying spectral properties of the information source designed in Section 6.2.2.

definition of the trellis states implies that the trellis diagram, and consequently the resulting finite-state diagram, are in Moore-type form, *i.e.* the labels of the branches entering a given a state are all identical.

When $L_{\max}$ is large, the finite-state transition diagram built according to the above presented method has a large number of states. Therefore, a finite-state representation of the FL constraints with a minimum number of states is preferable. While minimizing the computational complexity of the resulting information source, the finite-state diagram with minimum number of states minimizes also the complexity of the design of finite-state encoders with the state-splitting algorithm. Such a representation, called the Shannon cover, can be obtained with the help of the Moore algorithm [79,75].

We illustrate the above presented methods for designing a trellis diagram and a Shannon cover which are matched to a given FL constraint. Table 6.3 shows that $\mathcal{L}=\{1101\}_{NRZI}$ is an eligible FL constraint. The constrained trellis

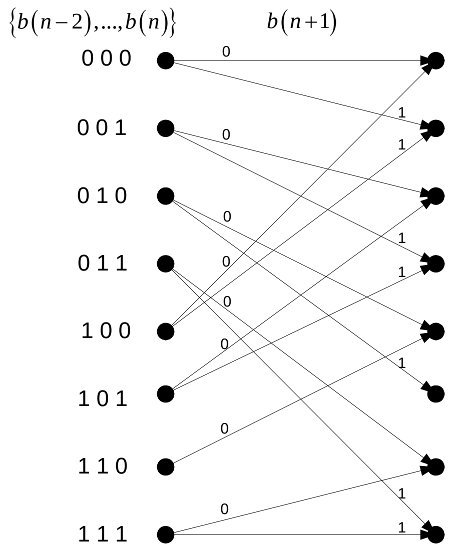

Figure 6.3: Trellis transition diagram for $\mathcal{L}=\{1101\}_{NRZI}$ FL constraint.

associated with $\mathcal{L}=\{1101\}_{NRZI}$ is shown in Figure 6.3. The application of Moore's algorithm to the finite-state diagram associated with the trellis diagram in Figure 6.3 results in a Shannon cover with four states, as shown in Figure 6.4. Note that both the trellis diagram and the Shannon cover, presented in Figure 6.3 and Figure 6.4 respectively, are in Moore-type form.

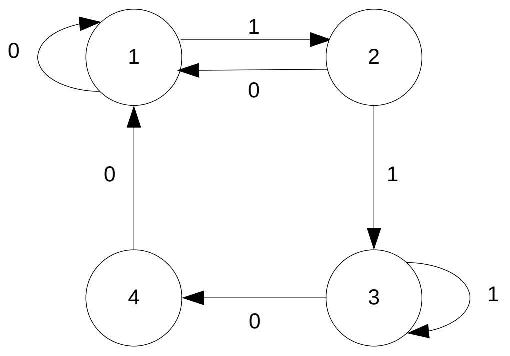

Figure 6.4: Shannon cover for $\mathcal{L}=\{1101\}_{NRZI}$ FL constraint.

The trellis of the Viterbi detector, called VD trellis[11], must be tuned according to the runlength constraints. A straightforward method for matching the VD trellis to the given FL constraint $\mathcal{L}=\{\underline{s}_1,\ldots,\underline{s}_M\}_{NRZI}$ is to consider $2^{M_{VD}-1}$ states where $M_{VD}=\max\left(N_g-1, L_{\max}+1\right)$, and N_g is the length of the GPR target $\underline{g}$. Since the VD trellis is used to detect bits $\hat{c}(n)$ in NRZ{-1,1} format, the VD trellis state transitions must be constrained according to the NRZ version $\mathcal{L}'=\{\underline{s}'_1,\ldots,\underline{s}'_{M'}\}_{NRZ}$ of the FL constraint $\mathcal{L}$. Note that the maximal length of the strings $\underline{s}'_j$, $j=1,\ldots,M'$, is now $L'_{\max}=L_{\max}+1$. The branches whose labels contain any of the strings $\underline{\tilde{s}}'_j$, $j=1,\ldots,M'$, where $\underline{\tilde{s}}'_j$ denotes

[11] Not to be confused with the trellis transition diagram introduced in Section 6.2.1.

the time reversed string associated with $\underline{s}_j'$, are removed. Such a VD trellis may be relatively computationally complex when $L_{\max}'$ is much larger than $N_g - 1$. Therefore, we shall design a more efficient VD trellis. We choose not to increase the size of the VD trellis, as compared to the uncoded case (*i.e.* $M_{VD} = N_g - 1$). At each VD trellis section, the Viterbi algorithm computes survivor branch metrics and updates the survivor path metrics. The VD trellis states are denoted $S_k(n) = \{c(n), ..., c(n - N_g + 2)\}$. For n sufficiently large, the tail of the survivor path at state $S_k(n)$ contains the string $\underline{s}' = [c(n - L_{\max}' + 1), ..., c(n)]$. If $\underline{s}' \in \mathcal{L}'$, then the survivor metric is set to ∞, so that the survivor path cannot be selected for detecting the data bits. This rule, which can be easily implemented, avoids increasing the number of states to build a constrained VD trellis.

6.2.2 Maxentropic Markov Source

Constrained sequences $\{b(n)\}$ in NRZI{0,1} format can be generated with the help of a suitably designed information source. The information source should maximize the information content of the generated constrained sequence $\{b(n)\}$. Such class of information sources are said to be maxentropic. Very efficient codes that can be designed for a given FL constraint [74] will have similar statistics as compared to a maxentropic information source for the same FL constraint. For this reason, we generate the constrained sequences with a maxentropic information source.

An information source can be represented by a labeled graph whose labeled edges are assigned transition probabilities. The constrained sequences are generated by reading off the labels in the graph. A Shannon cover for a given FL constraint (see Section 6.2.1) is used as the underlying graph of the information source. Let $\Sigma = \{S_1, ..., S_N\}$ represent the N states of the Shannon cover. The state transition probabilities are defined by the transition probability matrix $Q = \left[q_{i,j}\right]_{i,j=1,...,N}$, where $q_{i,j}$ is the transition probability from state S_i to state S_j. Let $\{Z_n\}$ represent the discrete-time stochastic process which describes the state sequence. This random process is a Markov chain (of order 1) [78] since the transition probabilities can be written as

$$q_{i,j} = \Pr\left[Z_n = S_i \middle| Z_{n-1} = S_j\right], \quad 1 \le i, j \le N . \tag{6.3}$$

Note that the transition probabilities $q_{i,j}$, $i, j = 1, ..., N$, are independent of the time index n. The information source is called a Markov source, since the random process $\{Z_n\}$ is Markovian.

When the transition probabilities are suitably designed, the Markov source maximizes the information content of the constrained sequences $\{b(n)\}$. We describe now how the information content, or entropy, is measured. Let $\{X_n\}$ represent the discrete-time stochastic process which describes the constrained sequences $\{b(n)\}$. Let ζ represent the function which associates a given state $S_k \in \Sigma$ to the label (in alphabet $\{0,1\}$) of the incoming edges. This function is well defined since the labeled graph is in Moore-type form. Thus, the sequence $\{X_n\}$ is given by $X_n = \zeta(Z_n)$, and is called the output of the Markov source. The entropy $H\left\{X\right\}$ of the Markov source is computed in two steps. First, we define the state entropies, and then we average the state entropies according to the steady state probabilities. Let $\left\{S_{k_1},...,S_{k_{n_k}}\right\}$ represent the one-step successors of state S_k, with $n_k = 1$ or 2, and $k = 1,...,N$. The Markov source is said to be 'unifilar' if the labels of the state transitions $\left(S_k, S_{k_i}\right)$ are different, with $i = 1,...,n_k$. In the context of constrained coding theory, the Markov source is said to be unifilar if its associated directed labeled graph is deterministic. A measure of the information contained in the bits generated from state S_k, called the entropy of state S_k, was proposed by Shannon [77]

$$H_k = H\left(q_{k,k_1},...,q_{k,k_{n_k}}\right) \triangleq -\sum_{i=1}^{n_k} q_{k,k_i} \log q_{k,k_i} \ . \tag{6.4}$$

Markov sources of practical interest are ergodic. Roughly speaking, a Markov source is said to be ergodic if from any state, any other state can be reached. In other words, the Markov source is ergodic if its underlying labeled graph is irreducible. Let $\tilde{P}_k^{(n)} = \Pr\left[Z_n = S_k\right]$ denote the probability of being at state S_k at time n. It can be easily shown that, the distribution of the state probabilities $\tilde{P}_k^{(n)}$ of the ergodic source reaches an equilibrium as $n \to \infty$, defined with the steady-state probabilities

$$\tilde{P}_k = \lim_{n\to\infty} \tilde{P}_k^{(n)} . \tag{6.5}$$

Shannon defined the entropy of the unifilar ergodic Markov source as the average of the state entropies H_k, $k = 1,...,N$, weighed in accordance with the steady-state probabilities $\tilde{P}_k$, $k = 1,...,N$

$$H\{X\} = \sum_{k=1}^{N} \tilde{P}_k H_k \, . \tag{6.6}$$

We wish to choose the transition probabilities $q_{k,j}$ in such a way that the entropy $H\{X\}$ is maximized. A Markov source with such transition probabilities is called maxentropic, and sequences generated by such a maxentropic unifilar source are called maxentropic sequences. Proved by Shannon, the maximum entropy of a unifilar Markov information source is given by

$$\mathcal{C} = \max_{q_{k,j}} H\{X\} = \log_2 \lambda_{\max} \, , \tag{6.7}$$

where $\lambda_{\max}$ is the largest real eigenvalue of the adjacency (or connection) matrix D. The existence of a positive eigenvalue and associate eigenvector with positive elements is guaranteed by the Perron-Frobenius theorem [76]. The state transition probabilities that maximize the entropy of the Markov source are

$$q_{k,j} = \lambda_{\max}^{-1} d_{k,j} \frac{p_j}{p_k}, \quad 1 \le k, j \le N \, , \tag{6.8}$$

where $[p_1, ..., p_N]^T$ is the eigenvector associated with the eigenvalue $\lambda_{\max}$.

6.3 Performance and Simulation Results

In this section, we first present an accurate expression for estimating the BER at the VD output. Thereafter, we investigate polynomial parity-check codes for simulating the post-processors.

6.3.1 BER Estimation

In order to identify the best FL constraints, the BER performance of the maxentropic coded PR channel is estimated. As proven in Appendix C, at medium to high SNR, a tight approximation of the BER is given by

$$P_b \approx \sum_{\underline{e} \in E} Q\left(\frac{d_e^2 + 2m_Y}{2\sigma_{Y'}} \right) \pi(e) w(e), \tag{6.9}$$

where $w(e)$ is the Hamming weight of the error event $\underline{e}$, $\pi(e)$ is the probability of the data patterns supporting the error event $\underline{e}$, d_e^2 is the Euclidean distance of the filtered error event $\underline{e}_g = \underline{e} \otimes \underline{g}$ with $\underline{g}$ representing the PR target, and m_Y and $\sigma_{Y'}$ are defined in Appendix C. In Eq. (6.9), $\sigma_{Y'}$ and $\pi(e)$ depend on the statistics of the FL constrained sequences. For accuracy, we shall derive exact expressions for the variables $\sigma_{Y'}$ and $\Pi(e)$ to the statistics of the maxentropic sequence $\{c(n)\}$.

As explained in Appendix C, $\sigma_{Y'}^2$ depends on the autocorrelation $\phi_{cc}(n,k) = E\left[c(n)c(n-k)\right]$ of the constrained data sequence $\{c(n)\}$. For identifying FL constraints, we have used NRZI format. However, for computing the autocorrelation $\phi_{cc}(n,k)$, it is more convenient to generate constrained sequences in NRZ format. Therefore, the FL constraint $\mathcal{L} = \{\underline{s}_1, \ldots, \underline{s}_M\}_{NRZI}$ is translated to its NRZ version $\mathcal{L}' = \{\underline{s}'_1, \ldots, \underline{s}'_{M'}\}_{NRZ}$. By applying the methods described in Section 6.2, a maxentropic Markov source for $\mathcal{L}'$ FL constraint can be built. When the source is chosen to be stationary, the sequence $\{c(n)\}$ is stationary and the autocorrelation is given by [78]

$$\phi_{cc}(n,k) = \phi_{cc}(k) = \underline{\zeta}^T \underline{\Pi} \underline{Q}^{|k|} \underline{\zeta}, \tag{6.10}$$

where $\underline{\zeta} = \left[\zeta(S_1),...,\zeta(S_N)\right]^T$, $\zeta(S_i)$ represents the generated bit $c(n)$ when the state S_i is visited, $\underline{\Pi} = diag\left(\tilde{P}_1,...,\tilde{P}_N\right)$, $\tilde{P}_i$, $i=1,...,N$, are the steady-state probabilities, and $\underline{Q}$ is the transition probability matrix.

The input data sequence $\{c(n)\}$ is stationary. As a result, the probability $\pi(e)$ of the constrained data patterns which support the error event e, can be viewed as the probability of the constrained data patterns which support an error string $\underline{\varepsilon}$ associated with the error event e. Further, the probability of the constrained data patterns which support $-\underline{\varepsilon}$ is equal to the probability of the constrained data patterns which support $\underline{\varepsilon}$. For this reason, we consider only the error string $\underline{\varepsilon}$ for which the first error element is +2. To build the error string, we choose to prefix and postfix $N_g - 1$ zeros to the error event $\underline{e}$. For instance, the error string associated with the error event $\underline{e} = \{+2,-2\}$ is $\underline{\varepsilon} = [0\ 0\ 0\ 0\ 2\ -2\ 0\ 0\ 0\ 0]$ of length $L_\varepsilon = L_e + 2N_g - 2$. Consequently, the probability $\pi(e)$ can be expressed as

$$\pi(e) = 2\sum_{\underline{c}\in C_\varepsilon} \Pr[\underline{c}],$$

where C_ε is the set of all constrained data patterns $\underline{c} = \left[c(1),...,c(L_\varepsilon)\right]$ that support the error string $\underline{\varepsilon}$. The probability $\Pr[\underline{c}]$ can be viewed as the probability $\Pr[\underline{S}]$ of the state sequences $\underline{S} \in \Sigma^{L_\varepsilon}$ such that $\zeta(\underline{S}) = \underline{c}$. Therefore, another formulation of $\pi(e)$ is given by

$$\pi(e) = 2\sum_{\underline{S}:\ \zeta(\underline{S})\in C_\varepsilon} \Pr[\underline{S}]$$

$$= 2 \sum_{\underline{S}:\ \zeta(\underline{S}) \in C_\varepsilon} \Pr\left[S_{i_1}, \dots, S_{i_{L_\varepsilon}} \right]$$

$$= 2 \sum_{\underline{S}:\ \zeta(\underline{S}) \in C_\varepsilon} \left[\prod_{j=L_\varepsilon-1}^{1} \Pr\left[S_{i_{j+1}} \big/ S_{i_j} \right] \right] \tilde{P}_{i_1}\,. \qquad (6.11)$$

When the variables $\sigma_{Y'}$ and $\pi(e)$ are matched to the statistics of the FL constrained sequences, using Eqns. (6.10) and (6.11), the BER approximation given in Eq.(6.9) is still valid, as shown in Figure 6.5. Similarly, Table 6.8 shows that the analytically estimated error event probabilities quite closely match the values obtained by simulations. Thus, we can confidently use the analytical means for estimating the probabilities needed for further design such as of the parity-based post-processor.

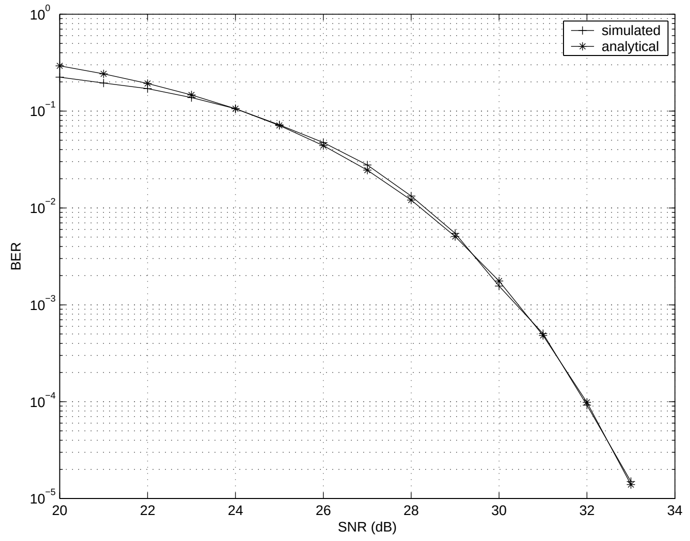

Figure 6.5: BER obtained by simulations and analytical expression for the $\mathcal{L} = \{1111111\}_{NRZI}$ maxentropic coded channel.

Table 6.8: Analytical and simulated error event probabilities for the maxentropic coded channel.

	$\underline{e}_1 = \{+2,-2\}$	$\underline{e}_2 = \{+2,-2,0,0,+2,-2\}$	$\underline{e}_3 = \{+2,-2,0,+2,-2\}$
Analytical	4.14e-5	1.87e-6	9.54e-7
Simulated	4.30e-5	1.85e-6	8.61e-7

6.3.2 Parity-Check Code

The insertion of parity bits can be done using one of the techniques mentioned in Section 3.3. Among these techniques, one makes the use of the unconstrained positions of the codewords [7433]. However, in this chapter, we do not insert explicitly the parity bits into the constrained sequence. Instead, parity bits are assigned to the maxentropic generated sequences and remembered at the receiver side[12]. We generate the parity bits using a polynomial cyclic code described in Section 4.2. For a given number of parity bits, the generator polynomial with minimum degree is easily obtained using simple computer programs and Matlab functions (*e.g. cyclpoly()*). We consider the polynomial codes which are able to detect the dominant error events. Obviously, we want as few parity bits as possible. Therefore, we start by considering single-parity codes. A single-parity linear cyclic code with generator polynomial $g(z)=1+z$ cannot detect any of the error events listed in Table 6.9. Also, a dual-parity linear cyclic code with generator polynomial $g(z)=1+z+z^2$ cannot detect the error event $\underline{e}_3$. Therefore, we consider a linear cyclic code with 3 and 4 parity bits, for which suitable generator polynomials are $1+z^2+z^3$ and $1+z^3+z^4$, respectively.

Selecting a suitable code rate is not straightforward. We choose the code rate R as follows. We first identify two integers p and q large enough which satisfy $p/q \le Cap(\mathcal{L})$. In order to take into account the decrease in code rate due to the use of a parity-check code, we choose the code rate as $R = p/(q+M)$, where M is the number of parity bits.

[12] That is, we assume that the parity bits are always correctly received at the receiver. Because of the relatively large codeword lengths which are considered, incorrectly receiving the parity bits should have negligible effect on the BER performance of the overall detector.

Table 6.9: Error event detection capability of two linear cyclic codes.

Dominant error events $\underline{e}_i$	$g(z)=1+z^2+z^3$	$g(z)=1+z^3+z^4$
$\underline{e}_1=\{+2,-2\}$	√	√
$\underline{e}_2=\{+2,-2,0,0,+2,-2\}$	√	√
$\underline{e}_3=\{+2,-2,0,+2,-2\}$	√	√
$\underline{e}_4=\{+2,-2,0,0,0,+2,-2\}$	√	√
$\underline{e}_5=\{+2,-2,0,+2,-2,0,+2,-2\}$	√	√
$\underline{e}_6=\{+2,-2,0,+2,-2,0,0,+2,-2\}$	√	√
$\underline{e}_7=\{+2,-2,0,0,+2,-2,0,+2,-2\}$	√	√
$\underline{e}_8=\{+2,-2,0,0,+2,-2,0,0,+2,-2\}$	√	√
$\underline{e}_9=\{+2,-2,0,+2,-2,0,+2,-2,0,+2,-2\}$	√	√
$\underline{e}_{10}=\{+2,-2,0,0,+2,-2,0,0,0,+2,-2\}$	√	√
$\underline{e}_{11}=\{+2,-2,0,0,0,+2,-2,0,0,+2,-2\}$	√	√
$\underline{e}_{12}=\{+2,-2,0,+2,-2,0,0,0,+2,-2\}$	√	√
$\underline{e}_{13}=\{+2,-2,0,0,0,+2,-2,0,+2,-2\}$	×	√

6.3.3 Simulation of MAP-based Post-Processor

In this subsection, we provide some simulation results for the MAP-based post-processor. The system under consideration is shown in Figure 6.6. The input data $\{b(n)\}$ is constrained by one of the FL constraints $\mathcal{L}_i$, $i=1,...,4$, given in Tables 6.5 and 6.7, and parity encoded with a linear cyclic code with a polynomial given in Table 6.9.

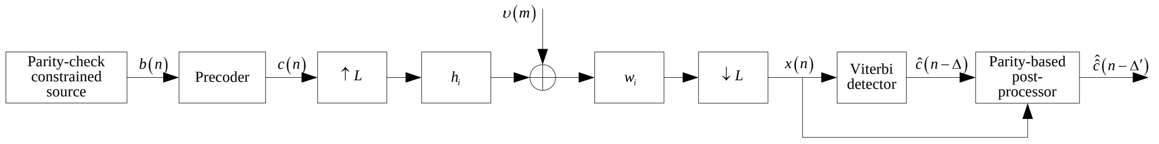

Figure 6.6: Schematic of a constrained parity-check coded perpendicular recording channel with Viterbi detector and parity-based post-processor.

Figure 6.7 shows the BER performance of constrained parity-check coded channels with MAP and ML post-processors, maxentropic constrained coded channel, and uncoded channel. The FL constraint is chosen to be $\mathcal{L}_3$, since it results in maximum separation between the two dominant error events. We choose to use the parity-check code with 4 parity bits (*i.e.* with generator polynomial $1+z^3+z^4$), since it supports the use of long parity codewords. Associated with the constraint $\mathcal{L}_3$ with capacity 0.9759,

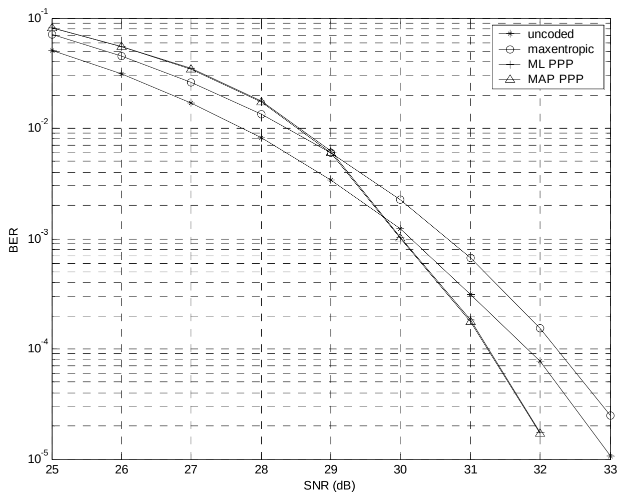

Figure 6.7: BER comparison of uncoded channel, $\mathcal{L}_3$ maxentropic coded channel, and constrained parity-check coded channels with MAP and ML-based post-processors.

the parity-check code can have the code rate $R = 288/(296+4) = 0.96$, which is 98.37% efficient. Figure 6.7 shows that, despite its high capacity, the FL constraint $\mathcal{L}_3$ brings a SNR loss of about 0.5dB at high SNR, as compared to the uncoded channel. The post-processors manage to gain about 1dB as compared to the uncoded channel. Despite our efforts to separate the probabilities between the dominant error events, the MAP-based and ML-based post-processors perform similar. Two reasons can account for this. First, since the noise at the detector input is not white, the post-processor is not optimal. The second reason may lie in the intrinsic structure of the dominant error events. Because of their similarities, the search range for FL constraints which increase the separation between dominant error event probabilities is narrowed. We believe that when the dominant error events do not show similarities, it is possible to separate significantly the probabilities without suffering from rate loss. Nevertheless, the fact that MAP and ML post-processors perform similar in recording channels is a blessing in disguise. This is because, implementation of MAP post-processor requires us to pre-compute the normalization constants. To do this computation, we need to have good knowledge of the channel and lot of theoretical work to obtain the required analytical expressions. On the other hand, such requirements are minimal in the ML-based post-processor.

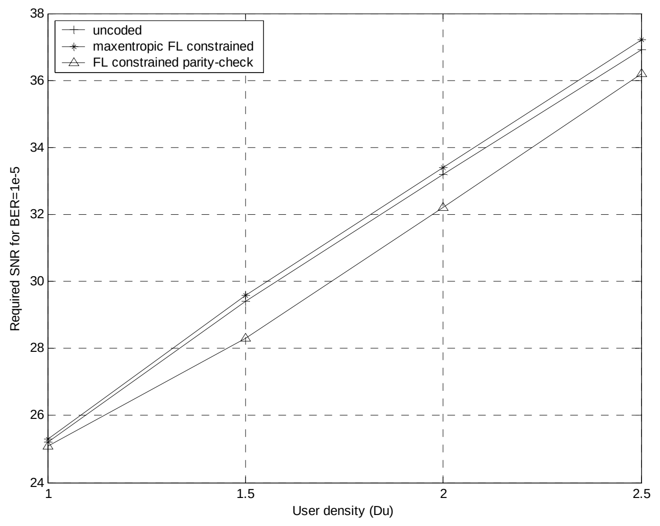

Figure 6.8: Required SNR for $BER = 10^{-5}$ versus user density for uncoded, maxentropic coded, and FL constrained parity-check coded channels.

Figure 6.8 shows the effect of density on the BER performance of Viterbi detector for different channel codes, by plotting the SNR required to achieve a BER of 10^{-5}

versus the user density. The plot ‘uncoded’ shows the detection performance for the uncoded channel. The plot ‘maxentropic FL constrained’ shows the detection performance for the channel whose input data is generated with a maxentropic source associated with a FL constraint. Obviously, as the density varies, the FL constraints that separate the probabilities of the dominant error event must be redesigned. At user densities $D_u = 1.0$, 1.5 and 2.5, a suitable constraint is $\mathcal{L} = \{1111101\}$ with capacity $Cap(\mathcal{L}) = 0.9883$. At $D_u = 2.0$, the chosen FL constraint is the FL constraint $\mathcal{L}_3$ given in Table 6.5. The simulations for the maxentropic constrained coded channel show little performance loss as compared to the uncoded channel. The plot ‘FL constrained parity-check’ shows the detection performance of the FL constrained source combined with a data-aided post processor. The parity-check code in use is the linear cyclic code with polynomial $1 + z^2 + z^3$. The BER simulations for the FL constrained parity-check coded channel show a SNR gain of about 1dB for a wide range of densities, as compared to the uncoded channel. The results obtained from Figure 6.8 are in contrast with those obtained from Figure 5.7 (Section 5.4), where the detection performance of the $MTR(1/2;6)$ block code and the parity-based post-processor suffer much from rate loss. In Figure 6.8, we can see clearly, that the effect of rate loss is minimized by considering high-capacity constraints.

6.4 Conclusions

In this chapter, we presented a method for designing very high capacity FL constraints which are matched to MAP-based post-processor by targeting dominant error events. The FL constraints separate the probabilities of dominant error events at the cost of rate loss. The number of possible FL constraints can be phenomenal. By constraining the search to forbidden strings with large fixed length, the search complexity is greatly reduced. As compared to integer-valued PR targets, monic-constrained GPR targets result in close-to-white noise at Viterbi detector input. Since this is in favor of the optimality of the post-processors, monic-constrained GPR targets are used. For simulating the BER performance of post-processors, maxentropic Markov source combined with linear cyclic codes are used. The overall system results in 1dB coding gain as compared to the uncoded case. However, the MAP-based and ML-based post-processors perform similar.

Chapter 7

Conclusions

In this thesis, we investigated the design of efficient constrained codes and parity-check codes for perpendicular recording channels.

7.1 Summary

Chapter 1 presented a very brief overview of magnetic data storage systems. Chapter 2 sets up the perpendicular magnetic recording channel which is used throughout this thesis. Chapter 3 elaborates on the topic of constrained codes and their application to improve detection performance in PRML receivers. Chapter 4 gives a detailed analysis (novel) of parity-based post-processing schemes. Chapter 5 presents the design of a novel parity-check code combined with a strong distance-enhancing modulation code for perpendicular recording channels. Chapter 6 elaborates on the original design of runlength constraints, for targeted error events, which are matched to MAP-based post-processors.

In this thesis, we have tackled three problems. The first and main problem is to derive the optimum receiver for post-processors based on MAP criterion. The second problem is to assess the effect of code rate on the performance of Viterbi detector and parity-based post-processor. The third problem is the design of general runlength constraints matched to MAP-based post-processors.

Firstly, most post-processors described in the literature can be viewed as multiple signals detector whose optimality is based on ML criterion. The work reported in this thesis (Chapter 4) shows that MAP-based post-processors may outperform ML-based post-processors when the error event probabilities are distinct. This has motivated the design of a novel parity-check code (Chapter 5) which brings 4.2dB coding gain in ideal recording channel.

Secondly, the performance of constrained codes is very sensitive to the code rate. With low code rates, distance-enhancing codes can result in large distance gain. However, the rate loss may overwhelm the distance gain. Therefore, for minimizing the rate loss, we have chosen to combine the constrained code and the parity-check code into one (Chapter 5). The use of low code rates tends to increase the correlation of the noise at VD input, which in turn makes the post-processor less optimal. With high rate, the performance of constrained codes suffers smaller rate loss. The post-processors also suffer less from noise colorization. But, the distance gain is smaller too.

Thirdly, the identification of general FL constraints which maximize the separation of the dominant error event probabilities is a problem with prohibitive complexity. We reduce the domain of search by considering deterministic FL constraints with a set of fixed-length forbidden strings. The proposed method shows that for high-capacity FL constraints, the separation between the dominant error event probabilities remains quite unchanged as compared to the uncoded case. At the cost of rate loss, the separation can be increased. For assessing the strength of the designed constraints, maxentropic information generate the required constrained sequences. Combined with a post-processor based on a linear cyclic code, the parity-check constrained system result in 1dB SNR gain as compared to the uncoded case. However, the use of the FL constraints is not effective in enhancing the performance of MAP-based post-processor as compared to ML-based post-processor. The main reason which accounts for this result may lie in the special structure of the set of dominant error events.

7.2 Directions for Further Work

There are several possible extensions which can be followed to make this research more complete.

First, one may study the effect of more practical noise, which includes jitter noise and transition noise, on the performance of constrained codes and parity-based post-processors.

Second, one may investigate analytically the failure rate of post-processors.

Third, one may extend the study of FL constraints to the general design of FL constraints for any given set of targeted error events with any given structure.

Bibliography

1. G.J. Tarnopolsky, “Hard disk drive at high magnetic areal density,” *IEEE Trans. Magn.*, vol. 41, no. 1, pp. 301-306, Jan. 2004.
2. R. Wood, Y. Sonobe, Z. Jin, and B. Wilson, “Perpendicular recording: The promise and the problems,” *J. Magn. Magn. Mater*., vol. 235, pp. 1-9, Oct. 2001.
3. R. Wood, “The feasibility of magnetic recording at 1 terabits per square inch,” *IEEE Trans. Magn.*, vol. 36, pp. 36–42, Jan. 2000.
4. P. R. Chevillat, E. Eleftheriou, and D. Maiwald, “Noise predictive partial-response equalizers and applications,” in *Proc. IEEE Intl. Conf. Commun.* (ICC), Jun. 1992, pp. 942–947.
5. 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.
6. K.A.S. Immink, “ Runlength-Limited Sequences,” *Proc. IEEE*, vol. 78, no. 11, pp. 1745-1759, Nov. 1990.
7. B.E. Moision, P.H. Siegel, and E. Soljanin, “Distance-enhancing codes for digital recording," *IEEE Trans. Magn.*, vol. 34, no. 1, pp. 69-74, Jan. 1998.
8. 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.
9. K. K. Fitzpatrick and C. S. Modlin, “Time-varying MTR codes for high density magnetic recording," in *Proc. IEEE Intl. Conf. Global Telecommun.* (GLOBECOM), Nov.1997, vol. 3, pp. 1250-1253.

10. B.H. Marcus, P.H. Siegel, J.K. Wolf, “Finite-state modulation codes for data storage,” *IEEE Trans. Commun.*, vol. 10, no. 1, pp. 5-37, Jan. 1992.

11. G.D. Forney Jr, “Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference,” *IEEE Trans. Inform. Theory*, vol. 18, no. 3, pp. 363 – 378, May 1972.

12. J.W.M. Bergmans, *Digital baseband transmission and recording*. Kluwer Academic Publishers, 1996, ch. 2, 6, 7.

13. R.D. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, and W. Schott, “A PRML system for digital magnetic recording,” *IEEE J. Selected Areas in Commun.*, vol. 10, no.1, pp. 38 – 56, Jan. 1992.

14. Y. Okamoto, H. Sumiyoshi, 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, pp. 2164–2166, Sep. 2000.

15. J. Moon and W. Zeng, “Equalization for maximum likelihood detectors,” *IEEE Trans. Magn.*, vol. 31, no. 2, pp. 1083–1088, Mar. 1995.

16. J.M. Cioffi, G.P. Dudevoir, M.V. Eyuboglu, and G.D. Forney, Jr., “MMSE decision-feedback equalizers and coding. I. equalization results,” *IEEE Trans. Magn.*, vol. 43, no. 10, pp. 2595-2604, Oct. 1995.

17. B.E. Moision, A. Orlitsky, and P.H. Siegel, “On codes that avoid specified differences,” *IEEE Trans. Inform. Theory*, vol. 47, no. 1, pp. 433-442, Jan. 2001.

18. P.M. Aziz, N. Sayiner, and V. Annampedu, “A rate 16/17 punctured MTR block code,” in *Proc. IEEE Intl. Conf. Commun.* (ICC), Jun. 1999, vol. 3, pp. 1643-1647.

19. S.A. Atlekar, M. Berggren, B.E. Moision, P.H. Siegel, and J.K. Wolf, "Error event characterization on partial response channels," *IEEE Trans. Inform. Theory*, vol. 45, no. 1, pp. 241-247, Jan. 1999.

20. R. L. Adler, D. Coppersmith, and M. Hassner, "Algorithms for sliding block codes - An application of symbolic dynamics to information theory," *IEEE Trans. Inform. Theory*, vol. 29, no. 1, pp. 5 – 22, Jan. 1983.

21. W.G. Bliss, "An 8/9 rate time-varying trellis code for high density magnetic recording," *IEEE Trans. Magn.*, vol. 33, no. 5, pp. 2746–2748, Sep. 1997.

22. T. Nishiya, K. Tsukano, T. Hirai, S. Mita, and T. Nara, "Rate 16/17 maximum transition run (3;11) code on an EEPRML channel with an error-correcting post-processor," *IEEE Trans. Magn.*, vol. 35, no. 5, pp. 4378-4386, Sept. 1999.

23. A.J. van Wijngaarden and K.A.S. Immink, "Combinatorial construction of high-rate runlength-limited codes," in *Proc. IEEE Intl. Conf. Telecommun.* (GLOBECOM), London, Nov. 1996, pp. 343-347.

24. J. Lee and V.K. Madisetti, "A rate 8/10 (0,6) MTR code and its encoder/decoder," *IEEE Trans. Magn.*, vol. 33, no. 1, pp. 2731–2733, Sep. 1997.

25. D.J. Costello, Jr., J. Hagenauer, H. Imai, and S.B. Wicker, "Applications of error control coding,“ *IEEE Trans. Inform. Theory*, vol. 44, no. 6, pp. 2531-2560, Oct. 1998.

26. K.A.S. Immink, P.H. Siegel, and J.K. Wolf, "Codes for digital recorders," *IEEE Trans. Inform. Theory,* vol. 44, no. 6, pp. 2260-2299, Oct. 1998.

27. K.D. Fisher, J.M. Cioffi, W.L. Abbot, P.S. Bednarz, and C.M. Melas, “An adaptive RAM-DFE for storage channels,” *IEEE Trans. Commun.*, vol. 39, no. 11, pp. 1559-1568, Nov. 1991.

28. T.M. Cover, “Enumerative source coding,” *IEEE Trans. Inform. Theory,* vol. 19, no.1, pp.73-77, Jan. 1973.

29. W. G. Bliss, “Circuitry for performing error correction calculations on baseband encoded data to eliminate error propagation,” *IBM Tech. Discl. Bull.,* vol. 23, pp. 4633-4634, 1981.

30. A.J. van Wijngaarden and E. Soljanin, "A combinatorial technique for constructing high rate MTR-RLL codes", *IEEE J. Selected Areas in Commun.*, vol. 19, no. 4, pp. 582-588, Apr. 2001.

31. B. Brickner and P. Padukone, “Partial response channel having MTR and parity constraints,” U.S. Patent 6388587 B1, May 2002.

32. S. Gopalaswamy and J.W.M. Bergmans, “Modified target and concatenated coding for d=1 constrained magnetic recording channels,” in *Proc. IEEE Intl. Conf. Commun*. (ICC), 2000, vol. 1, pp. 89-93.

33. A.J. van Wijngaarden and K.A.S. Immink, "Maximum runlength-limited codes with error control capabilities", *IEEE J. Selected Areas in Commun.*, vol. 19, no. 4, pp. 602-611, Apr. 2001.

34. R.D. Cideciyan and E. Eleftheriou, “Codes satisfying maximum transition run and parity-check constraints,” in *Proc. IEEE Intl. Conf. Commun.* (ICC), vol. 27, no. 1, Paris, France, Jun. 2004, pp. 602-611.

35. T. Conway, “A new target response with parity coding for high density magnetic recording channels,” *IEEE Trans. Magn.*, vol. 34, no. 4, pp. 2382-2386, Jul. 1998.

36. H. Sawaguchi, M. Kondou, N. Kobayashi, and S. Mita, “Concatenated error correction coding for higher-order PRML channels,” *in Proc. IEEE Intl. Conf. Global Telecommun.* (GLOBECOM), Sydney, Australia, Nov. 1998, pp. 2694-2699.

37. H. Sawaguchi, S. Mita, and J.K. Wolf, “A concatenated coding technique for partial response channels,” *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 695-703, Mar. 2001.

38. K. Saeki and M. Palmer, “Method and circuit for including parity bits in write data of a mass data storage device, or the like, using 48/54 MTR(3;k) code constraint, and post-processing circuit and method for processing readback data that includes said code constraint,” U.S. Patent 6581184 B1, Jun. 2003.

39. S. Lin and D.J. Costello, *Error Control Coding Fundamentals and Applications*. Prentice Hall, 1983, ch. 4.

40. W. Feng, A. Vityaev, G. Burd, and N. Nazari, “On the performance of parity codes in magnetic recording systems,” in *Proc. IEEE Intl. Conf. Telecommun.* (GLOBECOM), Nov. 2000, pp. 1877-1881.

41. B. Bloodworth, P. Siniscalchi, G. De Veirman, A. Jezdic, R. Pierson, and R. Sundararaman, “A 450 Mb/s analog front-end for PRML read channels,” in *Proc. IEEE Intl. Conf. Solid-State Circuits* (ISSCC), Feb. 1999, pp. 34-35.

42. A. Taratorin, D. Cheng, and E. Marinero, “Media noise, non-linear distortions, and thermal stability in high density recording,” *IEEE Trans. Magn*., vol. 36, no. 1, pp. 80-85, Jan. 2000.

43. R.D. Cideciyan, E. Eleftheriou, and T. Mittelholzer, “Perpendicular and longitudinal recording: A signal processing and coding perspective,” *IEEE Trans. Magn.*, vol. 38, no. 4, pp. 1698-1704, Jul. 2002.

44. H. Shafiee, “Timing recovery for sampling detectors in digital magnetic recording,” in *Proc. IEEE Intl. Conf. Commun.* (ICC), Dallas, Texas, Jun. 1997, pp. 577-581.

45. J.G. Proakis, “Equalization techniques for high-density magnetic recording,” *IEEE Signal Processing Magazine*, vol. 15, no. 4, pp. 73-82, Jul. 1998.

46. G.D. Forney, Jr., “The Viterbi algorithm,” *Proc. IEEE*, vol. 61, no. 3, pp. 268-278, Mar. 1973.

47. J.W.M. Bergmans, S.A. Rajput, and F.A.M. Van de Laar, “On the use of decision feedback for simplifying the Viterbi detector,” *Philips J. Research*, vol. 42, no. 4, pp. 399-428, Jan. 1987.

48. J. Moon and L.R. Carley, “Performance comparison of detection methods in magnetic recording,” *IEEE Trans. Magn.*, vol. 26, no. 6, pp. 3155-3172, Nov. 1990.

49. C.A. Belfiore and J.H. Park, “Decision feedback equalization,” *Proc. IEEE,* vol. 67, no. 8, pp. 1143-1156, Aug. 1979.

50. J. Moon, “The role of SP in data storage,” *IEEE Signal Processing Magazine,* vol. 15, no. 4, pp. 54-72, Jul. 1998.

51. P.H. Siegel and J.K. Wolf, "Modulation and coding for information storage," *IEEE Commun. Magazine*, vol. 29, no. 12, pp. 68-86, Dec. 1991.

52. R.D. Cideciyan, E. Eleftheriou, B.H. Marcus, and D.S. Modha, "Maximum transition run codes for generalized partial response channels," *IEEE J. Selected Areas in Commun.*, vol. 19, no. 4, pp. 619-634, Apr. 2001.

53. R.D. Cideciyan, J.D. Coker, E. Eleftheriou, and R.L. Gabriath, "Noise predictive maximum likelihood detection combined with parity-based post-processing," *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 714-720, Mar. 2001.

54. R. Karabed, P.H. Siegel, E. Soljanin, "Constrained coding for binary channels with high intersymbol interference," *IEEE Trans. Inform. Theory*, vol. 45, no. 6, pp. 1777-1797, Sep. 1999.

55. Y. Okamoto, N. Masunari, H. Yamamoto, H. Osawa, H. Saito, H. Muraoka, and Y. Nakamura, "Jitter-like noise cancellation using AR model of PR channel in perpendicular magnetic recording," *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2349-2351, Sep. 2002.

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

57. E.A. Lee and D.G. Messerchmidt, *Digital Communication*. 2nd edition, Kluwer Academic Publ., 1993, ch. 9,10.

58. H.K. Thapar and A.M. Patel, "A class of partial response systems for increasing the storage density in magnetic recording," *IEEE Trans. Magn.*, vol. 23, no. 5, pp. 3666-3668, Sep. 1987.

59. I. Lee and J.M. Cioffi, “Equalized maximum-likelihood receiver with a unit energy constraint,” *IEEE Trans. Magn.*, vol. 33, no. 1, pp. 855-862, Jan. 1997.

60. H. Sawaguchi, Y. Nishida, H. Takano, and H. Aoi, “Performance analysis of modified PRML channels for perpendicular recording systems,” *J. Magn. and Magn. Mater.*, vol. 235, no. 1-3, pp. 265-272, Oct. 2001.

61. P. Kovintavewat, I. Ozgunes, E. Kurtas, J.R. Barry, and S.W. McLaughlin, “Generalized partial-response targets for perpendicular recording with jitter noise,” *IEEE Trans. Magn.*, vol. 38, no. 5, pp. 2340-2342, Sep. 2002.

62. L. Chen and G. Mathew, “Analytical solution for optimum partial response target in Viterbi based receivers,” under review with *IEEE Trans. Commun.*, Jun. 2004.

63. 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.

64. A.D. Weathers, S.A. Altekar, and J.K. Wolf, “Distance spectra of PRML channels,” *IEEE Trans. Magn.*, vol. 33, no. 5, pp. 2809-2811, Sep. 1997.

65. B. Brickner and J. Moon, “Design of a rate 6/7 maximum transition code,” *IEEE Trans. Magn.*, vol. 33. no. 5, pp. 2749-2751, Sep. 1997.

66. B. Nikolic, M. Leung, and L. Fu, “A rate 8/9 sliding block trellis code with stationary detector for magnetic recording,” in *Proc. IEEE Intl. Conf. Commun.* (ICC), Jun. 1999, pp. 1653-1657.

67. J.J. Ashley, R. Karabed, and P.H. Siegel, “Complexity and sliding-block decodability,” *IEEE Trans. Inform. Theory*, vol. 42, no. 6, pp. 1925-1947, Nov. 1996.

68. E. Soljanin and A. Wijngaarden, “Application of distance enhancing codes,” *IEEE Trans. Magn.*, vol. 37, no. 2, pp. 762-767, Mar. 2001.

69. K.A.S. Immink, “A practical method for approaching the channel capacity of constrained channels,” *IEEE Trans. Inform. Theory*, vol. 43, no. 5, pp. 1389-1399, Sep. 1997.

70. J. Moon, “Signal-to-noise ratio definition for magnetic recording channels with transition noise,” *IEEE Trans. Magn.*, vol. 36, no. 5, pp. 3881-3883, Sep. 2000.

71. P. Chaichanavong and B.H. Marcus, “Optimal block-type decodable encoders for constrained systems,” *IEEE Trans. Inform. Theory*, vol. 49, no. 5, pp. 1231-1250, May 2003.

72. P. Franaszek, “Sequence-state coding for digital transmission,” *Bell Syst. Tech. J.*, vol. 47, pp. 113-157, 1968.

73. J. Campello de Souza, B.H. Marcus, R. New, and B.A. Wilson, “Constrained systems with unconstrained positions,” *IEEE Trans. Inform. Theory*, vol. 48, no. 4, pp. 866-879, Apr. 2002.

74. M.P. Béal, M. Crochemore, and G. Fici, “Presentations of constrained systems with unconstrained positions,” preprint IGM 2003-14, Jun. 2004. (http://www-igm.univ-mlv.fr/~beal/)

75. B.H. Marcus, R.M. Roth, and P.H. Siegel, *Handbook of coding theory*, V.S. Pless and W.C. Huffman, Elsevier, Amsterdam, 1998, ch. 20.

76. E. Seneta, *Non-negative matrices and Markov chains*, 2nd edition Springer, New York, 1980, ch. 1.

77. C.E. Shannon, “A mathematical theory of communication,” *Bell Syst. Tech. J.*, vol. 27, pp. 379-423, Jul. 1948.

78. K.A.S. Immink, *Codes for mass data storage*, Geldrop, Shannon Foundation Publishers, The Netherlands, 1999, ch. 5.

79. E.F. Moore, *Gedanken-experiments on sequential machines, Automata studies*, Princeton University Press, Princeton, New Jersey, 1956, pp. 129-153.

80. J. Moon and W. Zeng, “Equalization for maximum likelihood detectors,” *IEEE Trans. Magn.*, vol. 31, no. 2, pp. 1083-1088, Mar. 1995.

81. Editor team, *Handbook of mathematics,* People’s Education Publisher, Beijing, 1979, ch. 6.

82. G.L. Cariolaro and G.P. Tronca, “Spectra of block coded digital signals,” *IEEE Trans. Commun.*, vol. 22, no.10, pp. 1555-1564, Oct. 1974.

83. T.V. Souvignier, M. Oberg, R.E. Swanson, and J.K. Wolf, “Turbo decoding for partial response channels,” *IEEE Trans. Magn.*, vol. 48, no. 8, pp. 1297-1308, Aug. 2000.

84. A.D. Weathers and J.K. Wolf, “A new rate 2/3 sliding block code for the (1,7) run-length constraints with the minimal number of encoder states,” *IEEE Trans. Inform. Theory*, vol. 37, no. 3, pp. 908-913, May 1991.

85. W. Coene, H. Pozidis, and J. Bergmans, “Run-length limited parity-check coding for transition shift errors in optical recording,” *Intl. Conf. Global Telecommun.* (GLOBECOM), San Antonio, Texas, USA, Nov. 2001, pp. 25-29.

86. K. Cai, V. Krachkovsky, and J. Bergmans, “Performance bounds for parity coded optical recording channels with d=1 constraint,” in *Proc. IEEE Intl. Conf. Commun.* (ICC), New Orleans, USA, Jun. 2000, vol. 1, pp. 89-93.

87. L. Chen and M.R. Elidrissi, Nov. 2003, “private communication” .

List of Publications

Our work has enabled us to publish and prepare 3 different papers:

- M.R. Elidrissi, G. Mathew, “Novel parity-check code and post-processor for perpendicular recording channels,” *in Proc. Nineth Intl. Conf. Commun. Systems* (ICCS), Singapore, Sep. 2004, pp. 495-499.
- M.R. Elidrissi, G. Mathew, “Novel parity-check code and post-processor for perpendicular recording channels,” to be submitted.
- M.R. Elidrissi, G. Mathew, “Perfomance evaluation of maxentropic coded magnetic recording channels with post-processor,” to be submitted.

Appendix A

Elements of the Theory of Constrained Codes and State-Splitting Algorithm

In this appendix, we first review some terminology, fundamental concepts and results from the theory of constrained codes. Then, we present the state-splitting algorithm and apply it to an example that is relevant to the work reported in this thesis. Most of the material given in this appendix on theory and algorithms are taken from [75]. Please see [75] for a more complete treatment.

A.1 Fundamental Concepts

When specifying code design algorithms, it is very useful and convenient to refer to directed labeled graph of constrained sequences. More precisely, a directed labeled graph $G = (V, E, L)$ consists of a finite set of states $V = V(G)$, a finite set of directed edges $E = E(G)$ and an edge labeling $L = L(G) : E \to \mathcal{A}$ that assigns to each edge $e \in E$ a symbol in a finite alphabet $\mathcal{A}$. In our study, we consider $\mathcal{A} = \{0,1\}$, since the input data (*i.e.* $\{b(n)\}$ in NRZI format) to the recording channel are binary. Each edge[13] $e \in E$ has an initial state $\sigma(e)$ and a final state $\tau(e)$. A path in G is a finite sequence of edges $\mathfrak{e}$ in G in which the initial state of an edge having a predecessor corresponds to the final state of that predecessor. A constrained system S is the set of all symbol strings generated by the labeling of paths in G. This system, sometimes denoted $S(G)$, is said to be 'presented by' G. When there is no ambiguity, a directed labeled graph may be called simply a 'graph'. The connections in the graph G are conveniently described by a $|V(G)| \times |V(G)|$ matrix A_G, called adjacency matrix, whose entry $(A_G)_{u,v}$ is the number of edges from state u to state v in G. A parameter of particular importance in the application of the state-splitting algorithm is the 'minimum out-degree' of a graph. The 'out-degree' of a state u in a graph G is the number of outgoing edges from that state. The minimum out-degree of a graph G is the smallest of all out-degrees of the states in that graph. For the design of rate p/q finite-state encoder, it is very useful to describe explicitly the set of distinct words in S of length q. The q^{th} power of G, denoted G^q, is the directed labeled graph with the same states as G. Each edge in G^q is associated with one path of length q in G, and is labeled by the q-block generated by that path. The adjacency matrix A_{G^q} of G^q satisfies $A_{G^q} = (A_G)^q$.

For the purpose of encoder construction, it is important to consider directed labeled graphs with special properties. The most fundamental property is defined as follows. A graph is 'deterministic' if the labels of outgoing edges from each state are

[13] In this appendix, the edges and the set of edges are denoted by e and E, respectively. These variables should not be confused with error events and set of error events, denoted with the same symbols in the chapters.

distinct. It is important that any constrained system can be presented by some deterministic labeled graph [75]. The weaker version of the deterministic property is referred to as 'finite anticipation'. A graph is said to have finite anticipation if there is an integer N such that any two paths of length $N+1$ with the same initial state and labeling must have the same initial edge. The anticipation of G refers to the smallest N for which this condition holds. Note that a deterministic graph is equivalent to a graph with zero anticipation. Weaker than the 'finite anticipation' property is the 'lossless' property. A graph is said to be lossless if any two distinct paths with the same initial state and final state have different labelings. Another useful property of directed labeled graphs is 'irreducibility'. A graph G is said to be irreducible if there is a path in G from any specified starting state $u \in V(G)$ to any specified destination state $v \in V(G)$.

After giving some useful notation and terminology, we present the finite-state coding theorem (binary case). An encoder usually takes the form of a synchronous finite-state machine (as shown in Figure 3.3, Chapter 3). More accurately, for a given constrained system S and a positive integer n, a (S,n)-encoder is a labeled graph $\tilde{G}$ for which i) each state has out-degree n, ii) $S(\tilde{G}) \subseteq S \;\; S(\tilde{G})$, and iii) $\tilde{G}$ is lossless. The finite-state coding theorem is stated as follows.

Finite-state Coding Theorem:

Let S be a constrained system. If $p/q \le Cap(S)$, where p and q are positive integers, then there exists a rate p/q finite-state $(S, 2^p)$-encoder with finite anticipation.

A.2 State-Splitting Algorithm

The state-splitting algorithm (or ACH algorithm), introduced by Adler, Coppersmith, and Hassner [20], is a powerful method for designing efficient finite-state encoders. The algorithm implements a constructive proof of the above mentioned finite-state coding theorem by providing a recipe for designing finite-state encoders. The algorithm steps are given in Figure A.1. For a complete understanding of the state-splitting algorithm, we

---

(1) Select a labeled graph and integers as follows:
  - (a) Find a deterministic labeled graph G (or more generally, a graph with finite anticipation) which presents the given constrained system S.
  - (b) Find the adjacency matrix A_G of G.
  - (c) Compute the capacity $Cap(S) = \log_2 \lambda(A_G)$.
  - (d) Select a desired code rate p/q satisfying

$$Cap(S) \geq \frac{p}{q}.$$

  (It is desirable to keep p and q relatively small for complexity reasons).

(2) Construct G^q.

(3) Use the Franaszek algorithm for finding an $\left(A_G^q, 2^p\right)$-approximate eigenvector $\underline{x}$.

(4) Eliminate all states u with $x_u = 0$ from G^q, and restrict to an irreducible sink H of the resulting graph. Restrict $\underline{x}$ to be indexed by the states of H.

(5) Iterate Steps (5a)-(5c) below until the labeled graph has minimum out-degree at least 2^p:
  - (a) Find a non-trivial $\underline{x}$-consistent partition of the edges in H.
  - (b) Find the $\underline{x}$-consistent splitting corresponding to this partition, creating a labeled graph H' and an approximate eigenvector $\underline{x}'$.
  - (c) Let $H \leftarrow H'$ and $\underline{x} \leftarrow \underline{x}'$.

(6) At each state of H, delete all but 2^p outgoing edges and tag the remaining edges with binary p-blocks, one for each outgoing edge. This gives a rate p/q finite-state $\left(S, 2^p\right)$-encoder for the constraint system S.

Figure A.1: State-splitting algorithm.

give detailed explanations for each step.

Given a deterministic presentation G of the given constraint system S, Steps (1a)-(1d) are trivial. In Step (2), a recursive approach is adopted for determining the q^{th} power of graph G. In other words, the graph G^{i+1} is constructed with the help of G^{i}, for $i=1,...,q-1$. In Step (3), a nonnegative vector $\underline{x}=\left[x_1,...,x_{|V(G)|}\right]^T$, which satisfies $A_G^q\underline{x}\geq 2^p\,\underline{x}$ component-wise and $\underline{x}\neq 0$, is identified. Such a vector is called a $\left(A_G^q,2^p\right)$-approximate eigenvector. The existence of eigenvectors is guaranteed by the Perron-Frobenius theory [76]. In practice, an effective method for identifying an $\left(A_G^q,2^p\right)$-approximate eigenvector is provided by the Franaszek algorithm [75] for computing an $\left(A,n\right)$-approximate eigenvector, where A is a non-negative integer square matrix and n a positive integer. In Step (4), states u corresponding to $x_u=0$ are eliminated by deleting all the incoming edges and outgoing edges. At this stage, the resulting graph may not be irreducible because of possible isolated states.

Because the state-splitting transformation performed in Steps (5a)-(5c) require the graph to be irreducible, we need to restrict the graph in Step (4) to one of its irreducible sink, defined as an irreducible subgraph H for which any edge which originates in H must also terminate in H. A proof for the existence of at least one irreducible sink for any graph is given in [75]. In Steps (5a)-(5c), the graph H and the approximate eigenvector $\underline{x}$ are iteratively updated by applying a state-splitting transformation. More precisely, in Step (5a), we start with identifying a state $u\in V\left(H\right)$ that defines a basic $\underline{x}$-consistent splitting. It is proven in [75] that $\underline{x}$ is not the all-1 vector because H is irreducible. It is also shown that the states u such that $x_u = \max_{v=1,\ldots,|V(G)|} x_v$ define a 'basic $\underline{x}$-consistent splitting'. Let E_u denote the set of outgoing edges from such a state u in H. A basic out-splitting at state u is determined by a partition $E_u = E_u^{(1)} \cup E_u^{(2)}$ of E_u into two disjoint sets. For Step (5b), this partition is said to be $\underline{x}$-consistent if

$$\sum_{e \in E_u^{(1)}} x_{\tau(e)} \geq 2^p y^{(1)} \quad \text{and} \quad \sum_{e \in E_u^{(2)}} x_{\tau(e)} \geq 2^p y^{(2)}, \tag{A.1}$$

where $y^{(1)}$ and $y^{(2)}$ are positive integers such that $y^{(1)} + y^{(2)} = x_u$. The out-splitting defined by this partition is called a basic $\underline{x}$-consistent splitting. The current graph H is transformed to a graph H' by replacing state u with 2 descendant states $u^{(1)}$ and $u^{(2)}$. The assignment of edges to the states $u^{(1)}$ and $u^{(2)}$ is done according to the partition $E_u = E_u^{(1)} \cup E_u^{(2)}$. Detailed explanations of the rules for assigning edges to the states $u^{(1)}$ and $u^{(2)}$ are available in [75]. The $\left(A_H, 2^p\right)$-approximate eigenvector $\underline{x}$ needs also to account for the out-splitting performed at state u. The vector $\underline{x}'$ defined by

$$x'_v = \begin{cases} x_v & \text{if } v \neq u \\ y^{(1)} & \text{if } v = u^{(1)} \\ y^{(2)} & \text{if } v = u^{(2)} \end{cases}, \tag{A.2}$$

is clearly an $\left(A_{H'}, 2^p\right)$-approximate eigenvector. In Step (5c), the graph H and the $\left(A_G^q, 2^p\right)$-approximate eigenvector $\underline{x}$ are updated.

Finally (Step (6)), when the minimum out-degree of the graph H is at least 2^p, a finite-state $\left(S, 2^p\right)$-encoder can be easily constructed by deleting excess outgoing edges and tagging the remaining outgoing edges with binary p-blocks, for each state. Note that the finite-state encoder can be simplified using state-merging procedures [75].

For illustration purpose, we shall apply the state-splitting algorithm for the design of a finite-state encoder which corresponds to a constrained system relevant to this thesis. In Chapter 6, we have considered forbidden list (FL) constraints that reduce the probabilities of the dominant error events. We have focused on high capacity constraints. Since the use of large integers p and q, which are required for high rates, make the finite-state encoder extremely complex (*i.e.* large number of states and large number of edges for each state), we choose to illustrate the state-splitting algorithm on constrained systems with lower capacities. Even though high-rate finite-state encoders are impractical, some results concerning their design will be given. Specifically, we design a finite-state encoder for the $\mathcal{F}=\{111\}_{NRZI}$ FL constraint, also known as $MTR(j=2)$ constraint. This constraint, which was initially designed for eliminating $\{+2,-2,+2\}$ error event, reduces the probabilities of the error events $\{+2,-2\}$, $\{+2,-2,0,0,+2,-2\}$ and $\{+2,-2,0,+2,-2\}$ by a factor of 16, 32 and 16, respectively. Thus, this constraint can be an eligible constraint for the work done in Chapter 6. A deterministic graph G for this constraint is given in Figure[14] A.2.

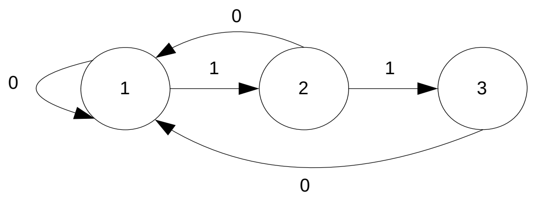

Figure A.2: Graph presenting the $\mathcal{F}=\{111\}_{NRZI}$ constrained system.

[14] This figure is the same as Figure 3.1 and is repeated here for convenience.

The capacity of the corresponding constrained system is 0.8791 (see Eq. (3.2)) obtained from the adjacency matrix

$$A_G = \begin{pmatrix} 1 & 1 & 0 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}.$$

According to the finite-state coding theorem, this capacity enables the design of a rate $7/8$ finite-state $\left(S, 2^7\right)$-encoder, which will be 99.5% efficient. The adjacency matrix of the graph G^8 is given by

$$A_{G^8} = \begin{pmatrix} 81 & 44 & 24 \\ 68 & 37 & 20 \\ 44 & 24 & 13 \end{pmatrix}.$$

The Franaszek algorithm gives the $\left(A_G^8, 2^7\right)$ -approximate eigenvector as $\underline{x} = \left[6, 5, 3\right]^T$. There is no state $u \in \{1, 2, 3\}$ such that $x_u = 0$ and G^8 is already irreducible (no element of A_{G^8} is zero). Let us detail the Steps (5a)-(5c) in the first iteration. State $u = 1$ defines a non-trivial $\underline{x}$-consistent partition $E_1 = E_1^{(1)} \cup E_1^{(2)}$, where $E_1^{(1)}$ contains 64 edges, and $E_1^{(2)}$ contains 17 edges. Also, the $\left(A_G^8, 2^7\right)$ -approximate eigenvector $\underline{x}$ satisfies

$$\sum_{e \in E_u^{(1)}} x_{\tau(e)} \geq 2^7 \mathrm{y}^{(1)} \quad \text{and} \quad \sum_{e \in E_u^{(2)}} x_{\tau(e)} \geq 2^7 y^{(2)}, \tag{A.3}$$

where $y^{(1)} = 3$, $y^{(2)} = 3$, and $y^{(1)} + y^{(2)} = 6 = x_1$. The adjacency matrix of the graph H' resulting from the $\underline{x}$-consistent splitting at state $u = 1$ is given by

$$A_{H'} = \begin{pmatrix} 64 & 64 & 0 & 0 \\ 17 & 17 & 44 & 24 \\ 68 & 68 & 37 & 20 \\ 44 & 44 & 24 & 13 \end{pmatrix}.$$

The descendants states, usually denoted $1^{(1)}$ and $1^{(2)}$, correspond to the states 1 and 2, respectively, of the graph H'. The corresponding $\left(A_{H'}, 2^7\right)$-approximated eigenvector is $\underline{x}' = [3,3,5,3]^T$. The graph H and the corresponding $\left(A_H, 2^7\right)$-approximated eigenvector $\underline{x}$ are updated. The out-degrees of states 1, 2, 3 and 4 are 128, 102, 193 and 125, respectively. Therefore, the minimum out-degree of the graph H is $102 < 2^7$. Steps (5a)-(5c) are reiterated until the minimum out-degree of H is at least 2^7. After 11 iterations, the graph H had 14 states and minimum out-degree equal to 2^7, which was reached at 11 states. Only three states (*i.e.* states 6, 11 and 14) had excess edges. Deleting judiciously the excess edges and tagging the edges with words of length 7 provides a rate $7/8$ $\left(S, 2^7\right)$-encoder.

Other finite-state encoders are also designed. We show in Table A.1 their basic characteristics. It is important to note that the initial $\left(A_G^q, 2^p\right)$-approximate eigenvector $\underline{x}$ characterizes the range of the number of states of the designed finite-state encoder. It is shown in [75] that the number of states N_s of the finite-state encoder satisfies

$$\max_{v=1,\dots,|V(G)|} x_v \leq N_s \leq \sum_{v=1}^{|V(G)|} x_v \,. \tag{A.4}$$

The results presented in Table A.1 show finite-state encoders for which the number of states reach the upper-bound. It is possible, through state-merging procedures [75], to reduce the number of states. However, the methods available cannot be applied to high-rate finite-state encoders, since they would require phenomenal computational complexity. It is clearly seen that finite-state encoders with extremely high efficiency require a very large number of states. The $MTR(j=3)$ constraint is an appropriate constraint for recording channels with dominant error event $\{+2,-2,+2\}$ (see Section 3.2.2). No code with code rate strictly larger than 16/17 has been designed. Several rate 16/17 $MTR(j=3)$ codes have been presented [22, 30, 18]. None of them has been designed with the ACH algorithm, because of complexity reasons. Even though it is impractical, we have designed here a rate 17/18 finite-state encoder for the $MTR(j=3)$ constraint. This example illustrates clearly the need for choosing p and q relatively small for designing codes with the state-splitting algorithm.

Table A.1: Features of finite-state encoders designed for various runlength constraints.

Constraint	Capacity	Code rate (p/q)	Efficiency (%)	Initial approximate eigenvector	Number of states
$MTR(j=2;k=8)$	0.8760	7/8	99.89	$[44,43,43,42,40,37,31,20,37,24]^T$	361
$\mathcal{F}=\{1101\}_{NRZI}$	0.90054	9/10	99.94	$[60,52,37,32]^T$	181
$\mathcal{F}=\{1101\}_{NRZI}$	0.90054	8/9	98.71	$[5,4,3,2]^T$	14
$MTR(j=3)$	0.94678	17/18	99.75	$[8,7,6,4]^T$	25
$MTR(j=3)$	0.94678	16/17	99.41	$[4,4,3,2]^T$	12
$MTR(j=3)$	0.94678	15/16	99.02	$[2,2,1,1]^T$	6

Appendix B

Energy of the Channel Bit Response

We evaluate in this appendix the energy of the perpendicular magnetic recording channel bit response. This computation is motivated by the analysis to examine the effect of code rate on the SNR at the output the recording channel (see Section 3.2.1).

The bit response of the recording channel is given by $h(t) = h_s(t) - h_s(t-T)$, where $h_s(t)$ is the channel step-response given in Eq. (2.4) which is recalled here for convenience

$$h_s(t) = \frac{A}{2}\tanh\left(\frac{\log(3)}{T_{50}}t\right) \tag{B.1}$$

The energy E_b of the bit response is given by

$$E_b = \int_{-\infty}^{\infty} h(t)^2\,dt = \frac{A^2}{4}\int_{-\infty}^{\infty}\left[\tanh\left(\frac{\log(3)}{T_{50}}t\right) - \tanh\left(\frac{\log(3)}{T_{50}}(t-T)\right)\right]^2 dt\,. \tag{B.2}$$

With the change of variable $x = \frac{\log(3)}{T_{50}}t$, Eq. (B.2) becomes

$$E_b = \frac{A^2}{4}\frac{T_{50}}{\log(3)}\int_{-\infty}^{\infty}\left[\tanh(x) - \tanh\left(x - \frac{\log(3)}{T_{50}}T\right)\right]^2 dx\,. \tag{B.3}$$

Let us define $\alpha = \frac{\log(3)}{T_{50}} T$, $\phi_\alpha(x) = \tanh x - \tanh(x-\alpha)$ and $I(\alpha) = \int_{-\infty}^{\infty} \phi_\alpha^2(x)\,dx$. Then, the energy E_b is given by

$$E_b = \frac{A^2 T}{4\alpha} I(\alpha) . \tag{B.4}$$

In order to compute $I(\alpha)$, we simplify $\phi_\alpha(x)$ as

$$\phi_\alpha(x) = \frac{e^{2x}-1}{e^{2x}+1} - \frac{e^{2x-2\alpha}-1}{e^{2x-2\alpha}+1} = \frac{2e^{2x-\alpha}(e^{\alpha}-e^{-\alpha})}{(e^{2x}+1)(e^{2x-2\alpha}+1)}$$

$$= \frac{\sinh(\alpha)}{\cosh(x)\cosh(x-\alpha)} = \frac{2\sinh(\alpha)}{\cosh(2x-\alpha)+\cosh(\alpha)} . \tag{B.5}$$

With the change of variable $u = x - \frac{\alpha}{2}$, the function $I(\alpha)$ in Eq. (B.4) is given by

$$I(\alpha) = \int_{-\infty}^{\infty} \left(\frac{2\sinh(\alpha)}{\cosh(2u)+\cosh(\alpha)} \right)^2 du = 2\int_{0}^{\infty} \left(\frac{2\sinh(\alpha)}{\cosh(2u)+\cosh(\alpha)} \right)^2 du$$

$$= 8\sinh^2(\alpha) \int_0^{\infty} \frac{4e^{4u}}{\left(e^{4u} + 2\cosh(\alpha)e^{2u}+1\right)^2} du . \tag{B.6}$$

With the change of variable defined as $v = e^{2u} + \cosh(\alpha)$, we get

$$I(\alpha) = 8\sinh^2(\alpha) \int_{1+\cosh(\alpha)}^{\infty} \frac{4(v-\cosh(\alpha))^2}{\left(v^2+1-\cosh^2(\alpha)\right)^2} \frac{dv}{2\left(v-\cosh(\alpha)\right)}$$

$$= 8\sinh^2(\alpha) \int_{1+\cosh(\alpha)}^{\infty} \frac{2v}{\left(v^2+1-\cosh^2(\alpha)\right)^2} dv$$

$$-16\sinh^2(\alpha)\cosh(\alpha) \int_{1+\cosh(\alpha)}^{\infty} \frac{1}{\left(v^2+1-\cosh^2(\alpha)\right)^2} dv . \tag{B.7}$$

Let us evaluate the two integrals on the right hand side (RHS) of Eq. (B.7) separately. The first integral is simplified as

$$\int_{1+\cosh(\alpha)}^{\infty} \frac{2v}{\left(v^2+1-\cosh^2(\alpha)\right)^2} dv = \left[\frac{-1}{v^2+1-\cosh^2(\alpha)}\right]_{1+\cosh(\alpha)}^{\infty} = \frac{1}{2\left(1+\cosh(\alpha)\right)}. \quad \text{(B.8)}$$

For the second integral, we use the following result [81]:

For the function $f(x) = \frac{1}{\left(ax^2+c\right)^n}$, we have

$$\int f(x)\,dx = \frac{x}{2c(n-1)(ax^2+c)^{n-1}} + \frac{2n-3}{2c(n-1)} \int \frac{dx}{(ax^2+c)^{n-1}}, \quad if \quad n \geq 2 \qquad \text{(B.9)}$$

$$\text{or} \quad \int f(x)\,dx = \frac{1}{2\sqrt{-ac}} \ln \frac{x\sqrt{a}-\sqrt{-c}}{x\sqrt{a}+\sqrt{-c}}, \qquad \text{if} \quad n=1, a>0, c<0. \qquad \text{(B.10)}$$

Comparing the $f(x)$ given above with the second integral in (B.7), we find that $a = 1 > 0$, $x = v$, $c = 1-\cosh^2(\alpha) < 0$ and $n = 2$. Therefore, using (B.9), we can evaluate the second integral in (B.7) as

$$\int_{1+\cosh(\alpha)}^{\infty} \frac{1}{\left(v^2+1-\cosh^2(\alpha)\right)^2} dv = \left[\frac{v}{2(1-\cosh^2(\alpha))\left(v^2+1-\cosh^2(\alpha)\right)}\right]_{1+\cosh(\alpha)}^{\infty}$$

$$+\left[\frac{1}{2\left(1-\cosh^2(\alpha)\right)} \frac{1}{2\sqrt{\cosh^2(\alpha)-1}} \ln \frac{v-\sqrt{\cosh^2(\alpha)-1}}{v+\sqrt{\cosh^2(\alpha)-1}}\right]_{1+\cosh(\alpha)}^{\infty}$$

$$= \frac{1}{4\sinh^2(\alpha)}\left[1+\frac{1}{\sinh(\alpha)}\log\frac{1+\cosh(\alpha)-\sinh(\alpha)}{1+\cosh(\alpha)+\sinh(\alpha)}\right]$$

$$= \frac{1}{4\sinh^2(\alpha)}\left(1-\frac{\alpha}{\sinh(\alpha)}\right). \qquad \text{(B.11)}$$

Substituting Eqns. (B.8) and (B.11) in (B.7), we get the function $I(\alpha)$ as

$$\begin{aligned} I(\alpha) &= 8\sinh^2(\alpha)\frac{1}{2\left(1+\cosh(\alpha)\right)} - 16\sinh^2(\alpha)\cosh(\alpha)\frac{1}{4\sinh^2(\alpha)}\left(1-\frac{\alpha}{\sinh(\alpha)}\right) \\ &= 4\left(\cosh(\alpha)-1\right) - 4\left(\cosh(\alpha)-\frac{\alpha}{\tanh(\alpha)}\right) \\ &= 4\left(\frac{\alpha}{\tanh(\alpha)}-1\right). \end{aligned} \tag{B.12}$$

Substituting Eq. (B.12) in (B.4), we get the energy of the bit response as

$$E_b = A^2 R T_u \left(\frac{1}{\tanh(\alpha)} - \frac{1}{\alpha}\right). \tag{B.13}$$

By using the third-order approximation $\tanh\alpha \approx \alpha - \frac{\alpha^3}{3}$, the energy of the bit response becomes

$$E_b \approx A^2 R T_u \frac{\alpha}{3} = \frac{\log(3)}{3}\frac{A^2 T_u}{D_u} R^2 . \tag{B.14}$$

Observe from Eq. (B.14) that the energy of the bit response is quadratically related to the code rate R. In other words, the signal to noise ratio at the channel output decreases in proportion to the square of the code rate.

Appendix C

Performance Analysis of Viterbi Detector

In the partial-response maximum-likelihood (PRML) receiver described in Chapter 2, the Viterbi detector (VD) is not an optimal implementation of maximum-likelihood sequence detection (MLSD), since the noise at its input is correlated due to the PR equalization process. Further, this noise also contains some amount of residual ISI. Therefore, in this appendix, we present a detailed performance analysis of the VD by taking into account the effects of noise correlation and mis-equalization. In this thesis, we need such a detailed and accurate analysis for two reasons. Firstly, accurate expression of the normalization constant is required for enhancing the performance of the post-processor based on maximum a posteriori (MAP) decision rule, which is implemented using a bank of error event matched filters. This post-processor is used in the simulations of Chapters 5 and 6. Secondly, in Chapter 6, accurate estimation of bit error rate (BER) and error event probabilities is required for determining the required runlength constraints. While the upper-bound given in Eq. (2.20) is the most accurate possible, its computation is extremely time-consuming because of the data-dependence of the argument of the $Q(.)$ function. This data-dependence is clearly seen in Eq. (2.23). Under some assumptions, which we will analyze in this appendix, the above mentioned data-dependence can be removed.

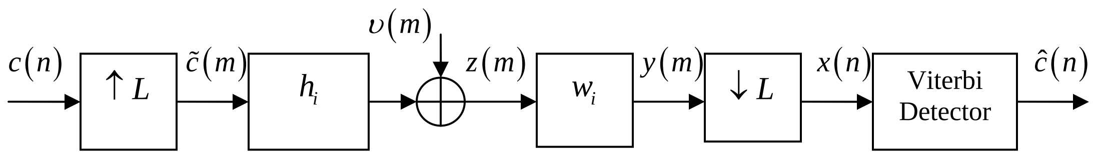

Figure C.1: Discrete-time model of the recording channel with PRML receiver.

Figure C.1 shows the structure of the PRML receiver under study in this thesis. The input data $c(n)$ denotes the user data in NRZ format with $c(n) \in \{-1,1\}$. The oversampled version $\{\tilde{c}(m)\}$ of $\{c(n)\}$ is defined by

$$\tilde{c}(m) = \begin{cases} c(n) & \text{if } m = nL \\ 0 & \text{if } m \neq nL, \end{cases} \tag{C.1}$$

where L is the oversampling factor. The channel noise $\upsilon(m)$, which models the electronics noise picked up by the read-head, is assumed to be white Gaussian with variance σ_υ^2 chosen according to Eq. (2.7). The read-back signal (*i.e.* the VD input) $\{x(n)\}$ is defined by $x(n) \triangleq y(nL+m_0)$, where m_0 is the delay from channel input $\tilde{c}(m)$ to equalizer output $y(m)$. The equalizer input is given by

$$z(m) = \underline{h}^T \underline{\tilde{c}}(m) + \upsilon(m), \tag{C.2}$$

where $\underline{h} = \left[h_0, \ldots, h_{N_h - 1}\right]^T$ is the T/L-spaced channel bit response with N_h taps, and $\underline{\tilde{c}}(m) = \left[\tilde{c}(m), \ldots, \tilde{c}(m - N_h + 1)\right]^T$. Thus, the VD input can be expressed as

$$x(n) = \underline{w}^T \underline{z}(nL+m_0) = \underline{w}^T \underline{\tilde{C}}(nL+m_0)\underline{h} + \underline{w}^T \underline{\upsilon}(nL+m_0), \tag{C.3}$$

where $\underline{w} = \left[w_0, ..., w_{N_w - 1} \right]^T$ is the T/L-spaced equalizer response with N_w taps, $\underline{\tilde{C}}(nL+m_0) = \left[\underline{\tilde{c}}(nL+m_0), ..., \underline{\tilde{c}}(nL+m_0-N_w+1) \right]^T$ is a $N_w \times N_h$ matrix, and $\underline{\upsilon}(nL+m_0) = \left[\upsilon(nL+m_0), ..., \upsilon(nL+m_0-N_w+1) \right]^T$. Eq. (C.3) can also be espressed as

$$x(n) = \sum_{i=0}^{N_{\tilde{f}}-1} \tilde{f}_i \tilde{c}(nL+m_0-i) + \eta_\upsilon(n) \tag{C.4}$$

where $\tilde{f}_i = h_i \otimes w_i$ is the T/L-spaced equalized channel response of length $N_{\tilde{f}} = N_h + N_w - 1$ and $\eta_\upsilon(n) = \underline{w}^T \underline{\upsilon}(nL+m_0)$ is the channel noise at the equalizer output (or, detector input). Using the definition of $\tilde{c}(m)$, we can rewrite (C.4) as

$$x(n) = \sum_{j=-j_1}^{j=j_2} \tilde{f}_{m_0+jL} \tilde{c}(nL-jL) + \eta_\upsilon(n) = \sum_{j=-j_1}^{j_2} f_j c(n-j) + \eta_\upsilon(n), \tag{C.5}$$

where $f_j = \tilde{f}_{m_0+jL}$, $j_1 = \left\lfloor \frac{m_0}{L} \right\rfloor$ and $j_2 = \left\lfloor \frac{N_{\tilde{f}} - 1 - m_0}{L} \right\rfloor$. Since the equalizer is expected to equalize the channel to the PR target $\underline{g} = \left[g_0, ..., g_{N_g-1} \right]^T$, we can express (C.5) as

$$x(n) = \sum_{i=0}^{N_g-1} g_i c(n-i) + \sum_{i=-j_1}^{j=j_2} f_i' c(n-i) + \eta_\upsilon(n)$$

$$= \underline{g}^T \underline{c}(n) + \underline{f}'^T \underline{c}'(n) + \eta_\upsilon(n), \tag{C.6}$$

where $\underline{c}(n) = \left[c(n), ..., c(n-N_g+1) \right]^T$, $\underline{f}' = \left[f'_{-j_1}, f'_{-j_1+1}, ..., f'_{j_2-1}, f'_{j_2} \right]^T$ is the residual ISI channel due to misequalization and $\underline{c}'(n) = \left[c(n+j_1), c(n+j_1-1), ..., c(n-j_2) \right]^T$ with

$$f_i' = \begin{cases} f_i & \text{if} \quad i \notin \{0,1,...,N_g-1\} \\ f_i - g_i & \text{if} \quad i \in \{0,1,...,N_g-1\}. \end{cases}$$

Eq. (C.5) can be rewritten as

$$x(n) = x_c(n) + \eta(n) \tag{C.7}$$

where $x_c(n) = \sum_{i=0}^{N_g-1} g_i c(n-i)$ is the signal component that the VD attempts to reconstruct, and $\eta(n) = \eta_c(n) + \eta_\upsilon(n)$ is the total noise component of $x(n)$ comprising a misequalization component $\eta_c(n) = \underline{f}'^T \underline{c}'(n)$ and a channel noise component $\eta_\upsilon(n)$. Note that the noise component $\eta_\upsilon(n)$ may be correlated because of the equalizer. Further, the residual ISI component $\eta_c(n)$ is non-Gaussian distributed. Thus, the total noise $\eta(n)$ at the VD input is actually non-Gaussian and correlated, unlike the most commonly used white noise assumption.

Let $\underline{c} = \left[c(0),...,c(N-1)\right]^T$ represent the actual input data sequence. The VD detects a sequence $\underline{\hat{c}} = \left[\hat{c}(0),...,\hat{c}(N-1)\right]^T$ according to the detection rule given by

$$\underline{\hat{c}} = \arg\min_{\underline{c}'} \left\| \underline{x} - \underline{x}_{c'} \right\|^2, \tag{C.8}$$

where $\underline{x} = \left[x(0),...,x(N+N_g-2)\right]^T$ and $\underline{x}_{c'} = \left[x_{c'}(0), x_{c'}(1),...,x_{c'}(N+N_g-2)\right]^T$ with $x_{c'}(n) = \sum_{i=0}^{N_g-1} g_i c'(n-i)$. An incorrect bit sequence $\underline{\hat{c}} \neq \underline{c}$ is detected if

$$\left\| \underline{x} - \underline{x}_{\hat{c}} \right\|^2 < \left\| \underline{x} - \underline{x}_c \right\|^2$$

$$\text{or} \qquad \left\| \underline{x}_c + \underline{\eta} - \underline{x}_{\hat{c}} \right\|^2 < \left\| \underline{\eta} \right\|^2, \tag{C.9}$$

where $\underline{x}_c$ and $\underline{x}_{\hat{c}}$ are defined similar to $\underline{x}_{c'}$. Eq. (C.9) can be expanded as

$$\sum_n \left(\sum_{i=0}^{N_g-1} g_i e_c(n-i) \right)^2 + 2\sum_n \sum_{i=0}^{N_g-1} g_i e_c(n-i)\eta(n) < 0 \tag{C.10}$$

where $e_c(n) = c(n) - \hat{c}(n)$ represents the error sequence. The probability of detecting $\underline{\hat{c}} \neq \underline{c}$ instead of $\underline{c}$ is given by

$$\Pr\left[\underline{\hat{c}}/\underline{c}\right] = \Pr\left[\frac{1}{2}\left\|e_g(n)\right\|^2 + \sum_n e_g(n)\eta(n) < 0\right] \tag{C.11}$$

where $e_g(n) = \sum_{i=0}^{N_g-1} g_i e_c(n-i)$.

Errors usually occur in the form of error events. If the error sequence $\{e_c(n)\}$ contains only a single error event $\underline{e}$, then Eq. (C.11) becomes the conditional probability of occurrence of error event $\underline{e}$ conditioned on the actual data sequence $\underline{c}$. From Eq. (C.11), we can derive the probability of error events and subsequently an easily computable upper-bound of the BER which is needed in Chapter 6. Further, Eq. (C.11) also serves for the derivation of the normalization constants for the post-processors under study in Chapter 4.

The probability P_e of an error event $\underline{e}$ starting at some time k_1 is given by [57]

$$P_e = \sum_{\underline{c}\in C_e} \Pr\left[e,\underline{c}\right] = \sum_{\underline{c}\in C_e} \Pr\left[e/\underline{c}\right]\Pr\left[\underline{c}\right] = \sum_{\underline{c}\in C_e} \Pr\left[\underline{\hat{c}}/\underline{c}\right]\Pr\left[\underline{c}\right], \tag{C.12}$$

where $\Pr\left[\underline{\hat{c}}/\underline{c}\right]$ is given in Eq. (C.11), $\Pr\left[\underline{c}\right]$ is the probability of the data sequence $\underline{c}$, and C_e is the set of all data sequences that support the error event $\underline{e}$ starting at time k_1. Note that the error event probability P_e depends on the starting time of the error event. The BER P_b is upper-bounded by

$$P_b \le \sum_{\underline{e} \in E} w(e) \sum_{\underline{c} \in C_e} \Pr\left[\underline{\hat{c}}/\underline{c}\right] \Pr\left[\underline{c}\right] \tag{C.13}$$

where E is the set of all possible error events, and $w(e)$ is the Hamming weight of the error event $\underline{e}$. At medium to high signal to noise ratio (SNR), the upper-bound in Eq. (C.13) becomes a good approximation of the BER.

Let $X = \frac{1}{2}\left\|e_g(n)\right\|^2 + \sum_n e_g(n)\eta(n) = \frac{d_e^2}{2} + \sum_n e_g(n)\eta_c(n) + \sum_n e_g(n)\eta_\upsilon(n)$ where $d_e^2 = \left\|\underline{e}_g\right\|^2$. For a given error event $\underline{e}$, we find that X is a random variable with a deterministic component, $\frac{d_e^2}{2} = \frac{1}{2}\left\|\underline{e}_g\right\|^2$, and a random component $Y = \sum_n e_g(n)\eta(n) = \sum_n e_g(n)\eta_c(n) + \sum_n e_g(n)\eta_\upsilon(n)$. Thus, the randomness in Y is due to the actual data sequence $\{c(n)\}$ and the channel noise sequence $\{\upsilon(m)\}$. Consequently, the evaluation of the probability $\Pr\left[\underline{\hat{c}}/\underline{c}\right]$ given by Eq. (C.11) depends on the different assumptions we make on the random component Y. Strictly speaking, Eq. (C.11) is being evaluated for a given data sequence $\underline{c}$. Therefore, Y has a mean given by $m_Y = \sum_n e_g(n)\eta_c(n)$. Since $\tilde{Y} = \sum_n e_g(n)\eta_\upsilon(n)$ is Gaussian distributed with zero mean, we get

$$\Pr\left[\underline{\hat{c}}/\underline{c}\right] = \Pr\left[\frac{d_e^2}{2} + Y < 0\right] = \Pr\left[\frac{d_e^2}{2} + m_Y + \tilde{Y} < 0\right]$$

$$= \Pr\left[\tilde{Y} > \frac{d_e^2}{2} + m_Y\right] = Q\left(\frac{d_e^2 + 2m_Y}{2\sigma_{\tilde{Y}}}\right), \tag{C.14}$$

where $Q(x) = \frac{1}{\sqrt{2\pi}}\int_x^\infty e^{-t^2/2}dt$, and $\sigma_{\tilde{Y}}^2$ is the variance of Y (or $\tilde{Y}$) given by

$$\sigma_{\tilde{Y}}^2 = E\left[\left(Y - m_Y\right)^2\right] = E\left[\tilde{Y}^2\right] = \sum_n \sum_k e_g(n) e_g(k) \phi_{\eta_\upsilon \eta_\upsilon}(n-k)$$

with $\phi_{\eta_\upsilon \eta_\upsilon}(n-k) = E\left[\eta_\upsilon(n)\eta_\upsilon(n-k)\right]$ being the autocorrelation of $\eta_\upsilon(n)$. Substituting Eq. (C.14) in Eq. (C.13), we get

$$P_b \le \sum_{\underline{e} \in E} w(e) \sum_{\underline{c} \in C_e} Q\left(\frac{d_e^2 + m_Y}{2\sigma_{\tilde{Y}}}\right) \Pr\left[\underline{c}\right]. \tag{C.15}$$

Because the mean m_Y is a function of the data $\underline{c}$, evaluation of the bound using Eq. (C.15) becomes cumbersome as we need to examine all possible data patterns for each error event. To simplify computations, a very commonly used assumption is to also treat the residual ISI $\eta_c(n)$ as Gaussian distributed with zero mean [80]. This is equivalent to considering the situation where the total noise $\eta(n) = \eta_c(n) + \eta_\upsilon(n)$ at the VD input is Gaussian with zero mean and variance $E\left[\eta_c^2(n)\right] + E\left[\eta_\upsilon^2(n)\right]$, and the residual ISI is zero. As a result, the random variable $Y = \sum_n e_g(n)\eta(n)$ becomes zero mean Gaussian with variance

$$\sigma_Y^2 = \sum_n \sum_k e_g(n) e_g(k) \phi_{\eta\eta}(n-k),$$

where $\phi_{\eta\eta}(n-k) = E\left[\eta(n)\eta(n-k)\right] = E\left[\eta_\upsilon(n)\eta_\upsilon(n-k)\right] + E\left[\eta_c(n)\eta_c(n-k)\right]$ is the autocorrelation of $\eta(n)$. Therefore, we get

$$\Pr\left[\hat{\underline{c}}/\underline{c}\right] = \Pr\left[\frac{d_e^2}{2} + Y < 0\right] = \Pr\left[Y > \frac{d_e^2}{2}\right]$$

$$= Q\left(\frac{d_e^2}{2\sigma_Y}\right). \tag{C.16}$$

Consequently, the BER P_b is upper-bounded by

$$P_b \le \sum_{\underline{e}\in E} w(e) \sum_{\underline{c}\in C_e} Q\left(\frac{d_e^2}{2\sigma_Y}\right)\Pr[\underline{c}] = \sum_{\underline{e}\in E} Q\left(\frac{d_e^2}{2\sigma_Y}\right)\pi(e)\,w(e), \tag{C.17}$$

where $\pi(e) = \sum_{\underline{c}\in C_e} \Pr[\underline{c}]$. Clearly, the Gaussian assumption on residual ISI results in a bound (C.17) that is much easier to evaluate compared to that in Eq. (C.15). But, it turns out that the upper-bound (C.17) is not tight enough at high SNR, as shown later in Figure C.2. Below, we present an approach [87] to circumvent this problem.

Let us rewrite the residual ISI as

$$\eta_c(n) = \sum_{i=-j_1}^{j_2} f_i' c(n-i) = \frac{1}{2}\sum_{i\in S_e} f_i' e_c(n-i) + \sum_{i\notin S_e} f_i' c(n-i), \tag{C.18}$$

where S_e is the set of all indices i from the set $\{-j_1, -j_1+1, \ldots, j_2\}$ such that $e_c(n-i) \ne 0$. Clearly, for a given error event, the first term on the RHS of Eq. (C.18) depends only on the error sequence $\{e_c(n)\}$ and is deterministic, whereas the second term is random with zero mean. If we assume that the second term in Eq. (C.18) is Gaussian, then the total noise $\eta(n)$ can now be looked upon (for a given error event) as Gaussian with mean $\frac{1}{2}\sum_{i\in S_e} f_i' e_c(n-i)$. As a result, $Y = \sum_n e_g(n)\eta(n)$ becomes Gaussian with mean m_e and variance $\sigma_{Y'}^2$ given by

$$m_e = \frac{1}{2}\sum_n e_g(n) \sum_{i\in S_e} f_i' e_c(n-i)$$

$$\sigma_{Y'}^2 = \sum_n \sum_k e_g(n) e_g(k) \left[\phi_{\eta_\upsilon\eta_\upsilon}(n-k) + \sum_{i\notin S_e}\sum_{j\notin S_e} f_i' f_j' \phi_{cc}(i-j)\right]$$

where $\phi_{cc}(k) = E\left[c(n)c(n-k)\right]$ is the input data autocorrelation. Using these, we get

$$\Pr\left[\underline{\hat{c}}/\underline{c}\right] = Q\left(\frac{d_e^2 + 2m_e}{\sigma_{Y'}}\right). \tag{C.19}$$

The resulting upper-bound of the BER can be expressed as

$$P_b \leq \sum_{\underline{e}\in E} Q\left(\frac{d_e^2 + 2m_e}{2\sigma_{Y'}}\right)\pi(e)\,w(e). \tag{C.20}$$

In order to examine the accuracies of the upper-bounds derived above, we did BER simulations for the perpendicular recording channel at user density Du=2.0, equalized to the monic constrained generalized partial response (GPR) target (designed for SNR=30dB). The input data is chosen to be uncoded. The results are shown in Figure C.2. The plots 'UB1' and 'UB2' correspond to the BER upper-bounds obtained according to Eq. (C.20) and Eq. (C.17), respectively, and the plot 'simulation' corresponds to the BER obtained by collecting at least 500 error bits for each SNR. Clearly, the upper-bound

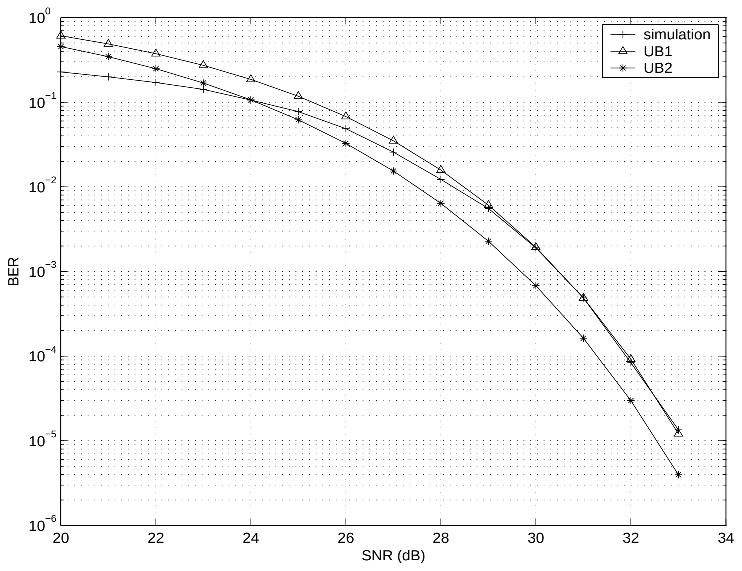

Figure C.2: BER obtained by simulation and two analytical upper-bounds.

given by Eq. (C.20) represents a very good approximation of the BER. On the other hand, the bound given in Eq. (C.17) is quite loose even though a large number of error events have been considered. Eq. (C.17) cannot provide an accurate upper-bound since the channel mis-equalization is significant and modeling it as Gaussian is highly inappropriate.

In order to verify whether the error event probabilities can be estimated accurately, simulations have been performed under the same conditions as mentioned above. We

Table C.1: Error event probabilities obtained by simulation and analysis.

	$\{+2,-2\}$	$\{+2,-2,0,0,+2,-2\}$	$\{+2,-2,0,+2,-2\}$
Probabilities from simulation	6.2661e-4	4.4558e-5	2.5367e-5
Analytical probabilities	6.6392.e-4	5.5676e-5	3.1508e-5

choose SNR=30dB. Table C.1 shows the error event probabilities obtained by collecting 10000 error events using simulations. The analytically computed probabilities are also shown in the table. The analytical approach used corresponds to that used for deriving Eq. (C.20). Observe that the analytical predictions match the simulations quite accurately.

Appendix D

BER Upper-Bound for the MTR(1/2;6) Block-Coded Channel

In this appendix, we present an analytical derivation of an upper-bound of the bit error rate (BER) for the block coded ideal channel described in Section 5.1. The commonly used upperbound (see Eq. (5.2)) assumes that the input data $c(n)$ is stationary. However, in Chapter 5, the input data is block coded. As a result, the data $c(n)$ is cyclostationary [82], with period equal to the length of the codewords. More precisely, the autocorrelation $\phi_{cc}(n,k) \triangleq E\left[c(n)c(n-k)\right]$ of the data $c(n)$ satisfies $\phi_{cc}(n,k) = \phi_{cc}(n-4,k)$ for any n and k. The expression for the BER upper-bound needs to take into account the statistics of the input data. For our derivation of the BER upper-bound, we follow the method presented in [57].

Figure D.1 shows the ideal channel with $MTR(1/2;6)$ block code and constrained Viterbi detector. The input of the detector is the summation of the encoded sequence $\underline{c}$ convolved with the target $\underline{g}$ and the colored noise $\underline{\upsilon}$. The Viterbi detector is matched to the constraints of the block code.

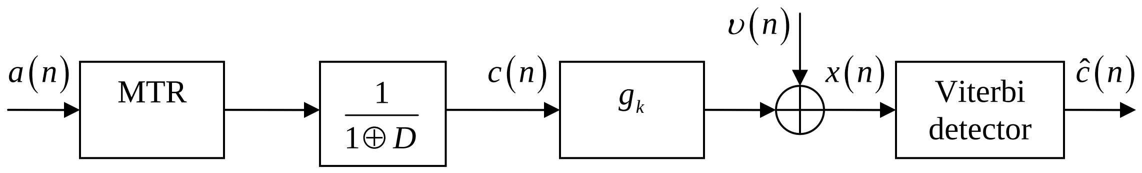

Figure D.1: Schematic of $MTR(1/2;6)$ constrained ideal recording channel with Viterbi detector.

Let E denote the set of all error events $\underline{e}$. Each error event causes one or more detection errors, where a detection error at time n means that the bit $c(n)$ at stage n of the trellis is incorrect. Let us define

$$\gamma_m(e) \triangleq \begin{cases} 1; & \textit{if } \underline{e} \textit{ has a detection error in position } m \textit{ (from the start)} \\ 0; & \textit{otherwise} \end{cases} .$$

This function characterizes the instants corresponding to detection errors in error event $\underline{e}$. The probability of a particular error event $\underline{e}$ starting at time i and causing a detection error at time n is

$$\gamma_{n-i}(e)\,\Pr[e,i].$$

Since the error events in E are disjoint (if one occurs no other can occur)

$$\Pr\left[\textit{detection error at time } n\right] = \sum_{i=-\infty}^{n} \sum_{e \in E} \gamma_{n-i}(e)\,\Pr[e,i].$$

Since the autocorrelation $\phi_{cc}(n,k)$ is cyclostationary with period 4, we can write $\Pr[e,i] = \mathrm{P}_{e,j}$ where $j = \mathrm{mod}(i,4)$. Also, the probability of detection error at time n is a periodic function of n with period 4. Consequently, we have

$$\Pr\left[detection\ error\ at\ time\ n\right] = \sum_{j=0}^{3} \sum_{\substack{i=-\infty \\ \mathrm{mod}(i,4)=j}}^{n} \sum_{e\in E} \gamma_{n-i}(e)\,\mathrm{P}_{e,j} \ .$$

Exchanging the order of summation, we get

$$\Pr\left[detection\ error\ at\ time\ n\right] = \sum_{j=0}^{3} \sum_{e\in E} P_{e,j} \sum_{\substack{i=-\infty \\ \mathrm{mod}(i,4)=j}}^{n} \gamma_{n-i}(e) \,.$$

Let us define $w_{n,j}(e) \triangleq \sum_{i:\ \mathrm{mod}(i,4)=j} \gamma_{n-i}(e)$, then

$$\Pr\left[detection\ error\ at\ time\ n\right] = \sum_{j=0}^{3} \sum_{e\in E} P_{e,j} w_{n,j}(e) \,. \tag{D.1}$$

Below, we shall derive an upper-bound for Eq. (D.1).

The probability that the error event $\underline{e}$ starts at time i such that $\mathrm{mod}(i,4) = j$ is

$$\mathrm{P}_{e,j} = \sum_{\underline{c}\in C_{e,j}} \Pr\left[\underline{\hat{c}}/\underline{c}\right] \Pr\left[\underline{c}\right],$$

where $\Pr\left[\underline{\hat{c}}/\underline{c}\right]$ is the probability of detecting $\underline{\hat{c}} = \left[\hat{c}(0),...,\hat{c}(4p-1)\right]$ when the transmitted sequence is $\underline{c} = \left[c(0),...,c(4p-1)\right]$, p is the number of MTR codewords, and $C_{e,j}$ is the set of sequences $\underline{c}$ supporting the error event $\underline{e}$ starting at a given time index in the j^{th} interleave. Since the noise sequence $\underline{\upsilon}$ is assumed white Gaussian, with variance σ^2, $\Pr\left[\underline{\hat{c}}/\underline{c}\right]$ can be upper-bounded by $Q\left(\frac{d_e}{2\sigma}\right)$, where $d_e \triangleq \left\| e \otimes \underline{g} \right\|$. Thus, the probability that the error event $\underline{e}$ starts at time i such that $\mathrm{mod}(i,4) = j$ satisfies

$$\mathrm{P}_{e,j} \le Q\left(\frac{d_e}{2\sigma}\right)\pi_j(e), \tag{D.2}$$

where $\pi_j(e) \triangleq \sum_{\underline{c}\in C_{e,j}} \Pr[\underline{c}]$ represents the probability of the data patterns $\underline{c}$ supporting the error event $\underline{e}$ starting at time i in the j^{th} interleave. Consequently, the upper bound for the probability of detection error at time n is given by

$$\Pr[\mathit{detection\ error\ at\ time\ n}] \le \sum_{\underline{e}\in E} Q\left(\frac{d_e}{2\sigma}\right)\sum_{j=0}^{3}\pi_j(e)\,w_{n,j}(e). \tag{D.3}$$

One detection error may cause more than one bit error. Thus,

$$\Pr[\mathit{bit\ error}] \le \Pr[\mathit{detection\ error\ at\ time\ n}].$$

$\Pr[\mathit{detection\ error\ at\ time\ n}]$ is a periodic function of the time index k with period 4. Consequently by averaging this probability over the 4 possible periods, we derive the probability of detection error:

$$\Pr[\mathit{detection\ error}] = \frac{1}{4}\sum_{i=0}^{3}\Pr[\mathit{detection\ error\ at\ time\ k} = 4m+i].$$

Under the pessimistic assumption $P_b = \Pr[\mathit{bit\ error}] \approx \Pr[\mathit{detection\ error}]$, we can write

$$\begin{aligned} P_b &\le \frac{1}{4}\sum_{i=0}^{3}\sum_{\underline{e}\in E} Q\left(\frac{d_e}{2\sigma}\right)\sum_{j=0}^{3}\pi_j(e)\,w_{4m+i,j}(e) \\ &\le \sum_{\underline{e}\in E} Q\left(\frac{d_e}{2\sigma}\right)\frac{1}{4}\sum_{i=0}^{3}\sum_{j=0}^{3}\pi_j(e)\,w_{4m+i,j}(e). \end{aligned} \tag{D.4}$$

Let us define $\underline{W}(e) \triangleq [w_{4m+i,j}(e)]_{0\le i,j\le 3}$ a 4×4 matrix, $\underline{\pi}(e) \triangleq [\pi_0(e),...,\pi_3(e)]^T$, and $\underline{t}(e) \triangleq \underline{W}(e)\underline{\pi}(e)$. Thus, we get

$$\sum_{i=0}^{3}\sum_{j=0}^{3}\pi_j(e)\,w_{4m+i,j}(e) = \sum_{i=0}^{3} t_i(e). \tag{D.5}$$

It can be easily shown that $\underline{W}(e)$ is a symmetric matrix. As a result, we get

$$\sum_{j=0}^{3} t_j(e) = \sum_{j=0}^{3} \pi_j(e) \sum_{i=0}^{3} w_{4m+i,j}(e) = \sum_{j=0}^{3} \pi_j(e) w(e), \tag{D.6}$$

where $w(e)$ is the Hamming weight of error event $\underline{e}$. Substituting Eqns. (D.5) and (D.6) in Eq. (D.4), we get

$$P_b \le \frac{1}{4} \sum_{\underline{e} \in E} Q\left(\frac{d_e}{2\sigma}\right)\left(\sum_{j=0}^{3} \pi_j(e)\right) w(e). \tag{D.7}$$

Note that the upper-bound (D.7) suggests that the probability $P_{e,j}$ that a given error event $\underline{e}$ starts at time index in the j^{th} interleave is upper-bounded by

$$P_{e,j} \le \frac{1}{4} Q\left(\frac{d_e}{2\sigma}\right) \pi_j(e). \tag{D.8}$$

Note also that the upper-bound (D.8) is more accurate than the upper-bound (D.2). This result will be verified with simulation results.

Table D.1: Dominant error events for the $MTR(1/2;6)$ coded ideal PR channel.

Dominant error events e_i	Squared Euclidean distances $d_{e_i}^2$
$\underline{e}_1 = \{+2, 0, -2\}$	72
$\underline{e}_2 = \{+2\}$	76
$\underline{e}_3 = \{+2, 0, -2, 0, +2\}$	76
$\underline{e}_4 = \{+2, 0, -2, 0, +2, 0, -2\}$	80
$\underline{e}_5 = \{+2, 0, -2, 0, +2, 0, -2, 0, +2\}$	84
$\underline{e}_6 = \{+2, 0, -2, 0, +2, 0, -2, 0, +2, 0, -2\}$	88
$\underline{e}_7 = \{+2, 0, -2, 0, +2, 0, -2, 0, +2, 0, -2, 0, +2\}$	92
$\underline{e}_8 = \{+2, 0, -2, 0, +2, 0, -2, 0, +2, 0, -2, 0, +2, 0, -2\}$	96
$\underline{e}_9 = \{+2, 0, -2, 0, +2, 0, -2, 0, +2, 0, -2, 0, +2, 0, -2, 0, +2\}$	100

In order to check the accuracy of this upper bound, we need a set of dominant error events and the corresponding probabilities $\pi_j(e)$. The dominant error events and their associated squared Euclidean distances d_e^2 are given in Table D.1. Let us simplify the expression for the probability $\pi_j(e)$. With $\underline{b}' = \left[b'(0),...,b'(4p-1)\right]$, the probability of the sequence $\underline{c}$ is $\Pr[\underline{c}] = \Pr[\underline{b}']$, where $c(n) = 2b'(n) - 1$. There is one-to-one relationship between the set of sequences $\underline{b} = \left[b(0),...,b(4p-1)\right]$ and the set of sequences $\underline{b}'$, when the choice of the first bit is deterministic. Consequently, $\Pr[\underline{b}'] = \Pr[\underline{b}]$. Note that $\Pr\left[b(4m),...,b(4m+3)\right] = \Pr\left[a(3m),...,a(3m+2)\right] = 2^{-3}$, $0 \le m \le p-1$, where $a(n)$ denotes the uncoded user data. Further, using the independence of the MTR codewords, we can write

$$\Pr[\underline{b}] = \prod_{m=0}^{p-1} \Pr\left[b(4m),...,b(4m+3)\right] = 2^{-3p}.$$

And hence, the probability $\pi_j(e)$ is given by

$$\pi_j(e) = N_j(e)\, 2^{-3p}, \qquad \text{(D.9)}$$

where $N_j(e)$ is the number of sequences $\underline{b}$ such that the corresponding sequences $\underline{c}$ satisfy $\underline{c} \in C_{e,j}$. The number $N_j(e)$ is evaluated using simple computer programs. The resulting probabilities $\pi_j(e)$ are given in Table D.2. Figure D.2 shows the simulation results. Figure D.2 gives the comparison of the BER obtained by simulations against the upper-bound given in Eq. (D.7).

Table D.2: Probabilities of the data patterns supporting the dominant error events starting at time k_1 such that $\mathrm{mod}(k_1, 4) = j$.

Error events	$j = 0$	$j = 1$	$j = 2$	$j = 3$
$\underline{e}_1$	1/4	¼	9/32	1/4
$\underline{e}_2$	3/4	¼	3/4	1/4
$\underline{e}_3$	3/32	¼	3/32	1/8
$\underline{e}_4$	1/32	1/8	9/256	1/8
$\underline{e}_5$	3/256	1/8	3/256	1/16
$\underline{e}_6$	1/128	3/32	33/4096	3/32
$\underline{e}_7$	13/4096	9/128	13/4096	1/16
$\underline{e}_8$	1/2048	1/16	9/16384	1/32
$\underline{e}_9$	3/16384	1/32	3/16384	1/32

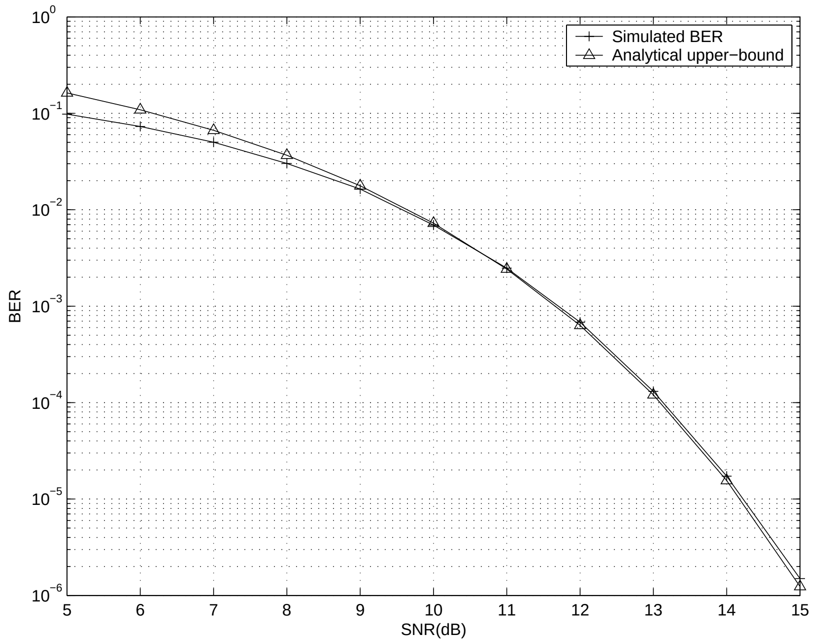

Figure D.2: Comparison of the analytical BER upper-bound and the simulated BER for the $MTR(1/2;6)$ coded ideal channel.

In order to verify whether the error event probabilities can be accurately estimated, we collected 30000 error events by simulations at SNR=12dB. Table D.3 shows the probabilities of the three dominant error events $\underline{e}_1$, $\underline{e}_2$ and $\underline{e}_3$, which are estimated using simulations as well as analytical computations according to Eq. (D.8).

Table D.3: Analytical and simulated error event probabilities for 3 dominant error events and for different starting time index.

	Analytical error event probabilities				Simulated error event probabilities			
	$j=0$	$j=1$	$j=2$	$j=3$	$j=0$	$j=1$	$j=2$	$j=3$
$\underline{e}_1$	2.47e-5	2.47e-5	2.78e-5	2.47e-5	2.29e-5	3.02e-5	2.61e-5	2.96e-5
$\underline{e}_2$	5.3e-5	1.77e-5	5.3e-5	1.77e-5	5.11e-5	2.53e-5	5.01e-5	2.63e-5
$\underline{e}_3$	6.62e-6	1.77e-5	6.62e-6	8.83e-6	5.88e-6	1.54e-5	6.19e-6	1.27e-5

Figure D.2 and Table D.2 show that the upper-bounds derived in this appendix for BER and error event probability are quite accurate and tight.