UNIVERSITY OF OKLAHOMA

GRADUATE COLLEGE

ADVANCED SIGNAL PROCESSING FOR MAGNETIC RECORDING ON PERPENDICULARLY MAGNETIZED MEDIA

A DISSERTATION

SUBMITTED TO THE GRADUATE FACULTY

in partial fulfillment of the requirements for the

Degree of

DOCTOR OF PHILOSOPHY

By

WU CHANG
Norman, Oklahoma
2010

ADVANCED SIGNAL PROCESSING FOR MAGNETIC RECORDING
ON PERPENDICULARLY MAGNETIZED MEDIA

A DISSERTATION APPROVED FOR THE
SCHOOL OF ELECTRICAL AND COMPUTER ENGINEERING

BY

Dr. J. R. Cruz, Chair

Dr. Lucy Lifschitz

Dr. Mark Yeary

Dr. Samuel Cheng

Dr. Tian-You Yu

To my wonderful wife Xingwei, and our lovely son Changchang

Acknowledgements

I would like to express my sincere gratitude to my advisor, Dr. J. R. Cruz, for his invaluable guidance, constant support and encouragement throughout my Ph.D. study and research. His expertise and wisdom bring me to a new level of research. His academic and non-academic advice and help make this work possible and will definitely benefit me forever.

I would also like to express my sincere appreciation to Dr. Lucy Lifschitz, Dr. Mark Yeary, Dr. Samuel Cheng and Dr. Tian-You Yu for their support and time serving on my doctoral committee.

I thank all the fellow students in the Communications Signal Processing Laboratory (CSPLab) for the helpful discussions and the friendship. In addition, I am appreciative of Dr. Richard M. Todd, Dr. Weijun Tan, Dr. Hongxin Song, who are former students of Dr. Cruz. I am really thankful to them for their help on my research.

I would like to thank Hitachi Global Storage Technologies (HGST) for providing a graduate fellowship, which made this work possible.

During my research and course work, I receive help from a number of faculty members and department staff. I thank every one of them, especially Ms. Lynn Hall, our graduate program assistant.

Finally, my deepest thanks go to my wonderful wife, Xingwei Wang and my lovely son, Kevin C. Chang. I also express my greatest gratitude to my parents and my sister. The love, support and patience from my whole family have been essential in seeing the completion of my Ph.D. program.

Table of Contents

List of Tables

List of Figures

Abstract

In magnetic recording channels (MRCs) the readback signal is corrupted by many kinds of impairments, such as electronic noise, media noise, intersymbol interference (ISI), inter-track interference (ITI) and different types of erasures. The growth in demand for the information storage, leads to the continuing pursuit of higher recording density, which enhances the impact of the noise contamination and makes the recovery of the user data from magnetic media more challenging. In this dissertation, we develop advanced signal processing techniques to mitigate these impairments in MRCs.

We focus on magnetic recording on perpendicularly magnetized media, from the state-of-the art continuous media to bit-patterned media, which is a possible choice for the next generation of products. We propose novel techniques for soft-input soft-output channel detection, soft iterative decoding of low-density parity-check (LDPC) codes as well as LDPC code designs for MRCs.

First we apply the optimal subblock-by-subblock detector (OBBD) to nonbinary LDPC coded perpendicular magnetic recording channels (PMRCs) and derive a symbol-based detector to do the turbo equalization exactly. Second, we propose improved belief-propagation (BP) decoders for both binary and nonbinary LDPC coded PMRCs, which provide significant gains over the standard BP decoder. Third, we introduce novel LDPC code design techniques to construct LDPC codes with fewer short cycles. Performance improvement is achieved by applying the new LDPC codes to PMRCs. Fourth, we do a substantial investigation on Reed-Solomon (RS) plus LDPC coded PMRCs. Finally, we continue our research on bit-patterned magnetic recording (BPMR)

channels at extremely high recording densities. A multi-track detection technique is proposed to mitigate the severe ITI in BPMR channels. The multi-track detection with both joint-track and two-dimensional (2D) equalization provide significant performance improvement compared to conventional equalization and detection methods.

1 Introduction to Magnetic Recording Systems

Today we are living in an information age. The need for information storage space is continuously increasing. Since the magnetic data storage systems are of low cost and high reliability, they have been the most popular high volume information storage systems. Among different kinds of magnetic recording systems, this dissertation focuses on magnetic hard disk drives (HDDs), though most of the techniques here can be applied to other magnetic recording systems straightforwardly.

The first HDD was developed by IBM in 1956, and contained 50 disks and provided a data capacity of 5MB [1]. The areal data density of this HDD was only 2Kb/in^2. After that people have never stopped inventing and applying new techniques to HDDs to increase areal data densities. The revolutionary improvements have been made on recording media, write heads, read heads, recording mode, signal processing techniques, error correction coding and many other aspects of the magnetic recording system. In the 1970s and 1980s, the annual growth of the areal density was about 30%. From the beginning of the 1990s, the areal density growth rate was boosted up to 60% per year, by employing magnetoresistive (MR) heads and a new design for the read channel, the so called partial-response maximum likelihood (PRML) detection channel. In the late 1990s, this incredible growth rate was continued with the introduction of an advanced MR head, the giant magnetoresistive (GMR) head. In recent years, the conventional magnetic recording mode, longitudinal magnetic recording (LMR), has been replaced by a new recording mode, namely perpendicular magnetic recording (PMR), which lead to the increase of the areal density at the pace of 30%~50% per year. Finally, the up to date

areal data density already surpassed 400Gb/in^2 [2].

The basic idea of digital magnetic recording is to change the magnetic field on the media according to the user data; the stored information can be retrieved later on by detecting the remnant field on the magnetic media. A write head is driven by the data signal current to magnetize the media, while a read head is needed to sense the magnetic flux change from the media and convert it back to signal current. In both write and read processes, the heads are flying above the media and keep a relative movement at some velocity. In LMR, the regions of magnetization directions of the bit regions on media are aligned horizontally, parallel to the surface of the disk and the movement track of the heads. By contrast, in PMR, the magnetic orientations of the bit regions are aligned vertically, perpendicular to the disk. Fig 1.1 illustrates these two recording modes.

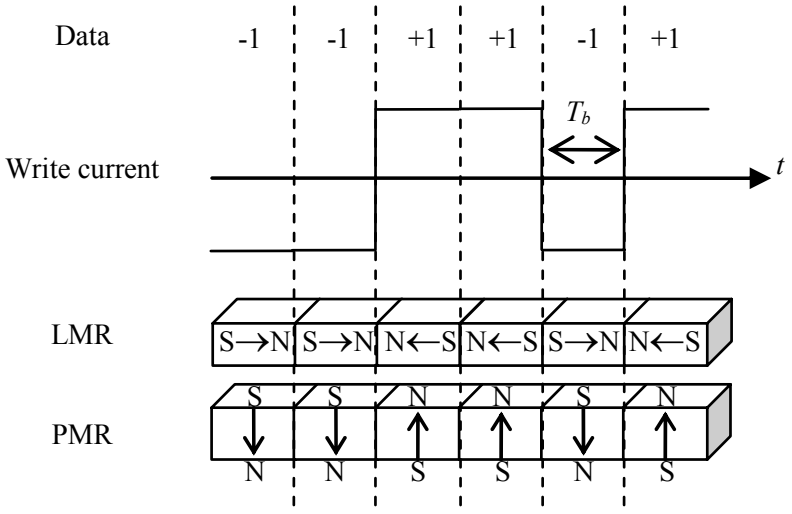


Fig. 1.1. Longitudinal and perpendicular magnetic recording.

In the first fifty years of the magnetic recording history of HDDs, LMR on continuous media was used exclusively. However, with the rapid increase in areal data density in the middle 2000s, this conventional recording technique approached its fundamental limit

due to its thermal instabilities, also called the superparamagnetic limit [3]. Finally, the highest areal data density achieved by LMR was around 100 Gb/in^2. As an alternative to LMR, PMR significantly expands the density limit. PMR has many advantages over LMR, including stronger recording field, thicker recording layers, no demagnetizing field in bit transitions, higher readback amplitude and so on. However, the current PMR technique also has its superparamagnetic limit. Actually, as early as 2000, people have predicted that the highest areal density that can be achieved by PMR is to be about 1Tb/in^2. This areal density limit is less than twice of what is currently in use, and the limit is expected to be reached in the near future. Then what kind of technology can push the recording density beyond 1Tb/in^2? So far the new techniques proposed include heat-assisted magnetic recording (HAMR) [4], bit-patterned magnetic recording (BPMR) [5] and two-dimensional magnetic recording (TDMR) [2], where BPMR is likely to be practically implemented in commercial products in the next few years. In bit-patterned media (BPM), as its name indicates, each bit is stored on an isolated perpendicularly magnetized island and the regions between islands are made of non-magnetic material. A simple illustration of a BPM HDD is given in Fig 1.2. Unlike the conventional continuous media, BPM is a radically redesigned media, which is capable of circumventing the superparamagnetic limit and eliminate the transition noise, a critical noise source in continuous media. In this dissertation, we make contributions on the signal processing of PMR, the current state-of-the-art technique as well as BPMR, the expected solution for the next generation of HDDs.

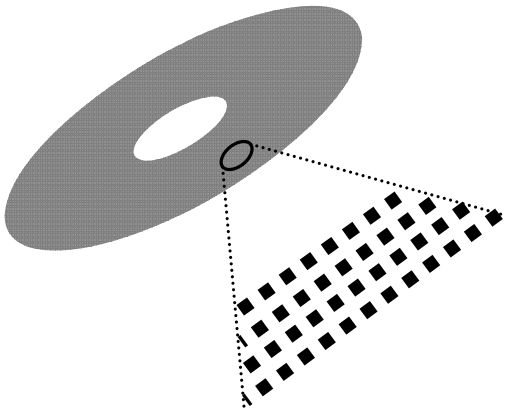


Fig. 1.2. Bit-patterned media: one bit per island.

1.1 Perpendicular magnetic recording channel

1.1.1 Channel response

In a magnetic recording channel (MRC), binary data (0 or 1) are recorded by magnetizing the storage media into two opposite directions. During the readback process, the read head senses the magnetic field of the storage media and outputs continuous time electrical signals according to the variation of the magnetization flux. Since there are only two magnetization directions on the media, flux changes are always caused by the switches of magnetization directions, or equivalently, the changes (0 →1 or 1 → 0) in the recorded binary sequence. Therefore, the readback process can be characterized by a 1 − D differential unit and the read head response to an isolated magnetization transition, $s(t)$, which corresponds to a single transition 0 →1 in the data sequence or −1 → 1 in the modulated bipolar sequences $a_k \in \{-1, 1\}$, at the input of the read channel. Shown in Fig. 1.3 is the linear model for the readback process, where the readback signal $r(t)$ is given by

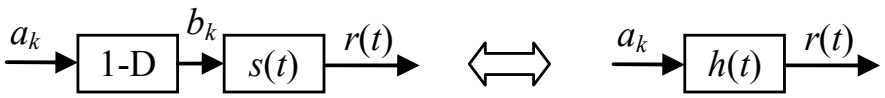


Fig. 1.3. Use dibit response to express the linear channel.

$$r(t) = \sum_k \left(a_k - a_{k-1}\right) s\left(t - kT_b\right) = \sum_k b_k s\left(t - kT_b\right), \tag{1.1}$$

where T_b is the bit interval and b_k is simply the differential (or transition) sequence of a_k. For PMR, the isolated *transition response* $s(t)$ is often modeled as a hyperbolic tangent function [6],

$$s(t) = V_p \tanh\left(\frac{\ln 3}{T_{50}} t\right), \tag{1.2}$$

where V_p is the half amplitude of the waveform and T_{50} is the time width required for $s(t)$ to rise from $-V_p/2$ to $V_p/2$, which is a fixed parameter for a given media. In the modeling of the read channel, it is useful to rewrite the readback signal $r(t)$ in (1.1) by absorbing the 1 – D unit into the continuous time waveform,

$$r(t) = \sum_k a_k h\left(t - kT_b\right), \tag{1.3}$$

where $h(t) = s(t) - s(t - T_b)$, is named *dibit response*. The dibit response is the channel response to two consecutive magnetization direction switches: a positive transition followed by a negative transition, or equivalently, the response to the binary input of unit impulse (Kronecker delta) sequence.

To measure how densely the data is recorded on the disk, a dimensionless parameter $D_c = T_{50}/T_b$ is defined as the recording density of the channel. Since T_{50} is a constant for a given media, higher recording density will cause more intersymbol interference (ISI).

Drawn in Fig. 1.4 are the isolated transition responses and dibit responses for recording density D_c = 1 and D_c = 2, where it is clear that increasing the recording density extends the span of waveforms over more bits and signal amplitude of the dibit response is attenuated due to the severe ISI.

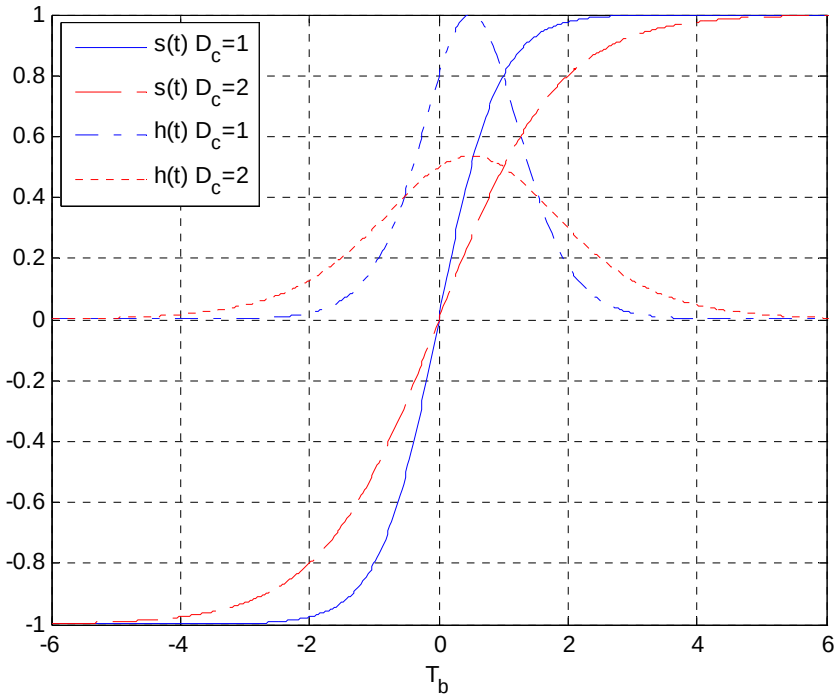


Fig. 1.4. Isolated transition responses and dibit responses of PMRCs.

1.1.2 Noises

In MRCs, the readback signal is corrupted by a variety of noise sources originated from the recording media, write and read heads, the head preamplifier and many other parts of the physical system. In this dissertation, we only consider electronic noise and a simple media noise component.

As in other communication systems, electronic noise (or thermal noise) $n(t)$ is always considered a noise component in the read channel of magnetic recording, which is

assumed to be an additive white Gaussian random process with single-sided power spectral density N_0.

Media noise is a dominant noise component in MRCs, which is caused by the granularity of the magnetic media. The transition noise is the major part of the media noise. Formed during the write process, the edges of the magnetic transitions are usually of zigzag patterns instead of clean and straight lines. In the subsequent read process, these imperfect transition boundaries lead to shifts of the readback signal position on the time axis and some variations on the width (T_{50}) of the transition response. In this dissertation, we only consider the time shift effect caused by transition noise on the readback signal, which is also called *position jitter noise*; all other noise components of media noise are ignored. The time shift of the readback signal is denoted by j_k and assumed to be a white Gaussian random sequence, which only happens where transitions occur. The readback signal corrupted by position jitter and electronic noises can be expressed as

$$r(t) = \sum_k b_k s\left(t - kT_b + j_k\right) + n(t). \tag{1.4}$$

However, this expression makes the simulation of the read channel very complicated. It will be more convenient if we can convert the contribution of j_k into additive terms. Accurately, the isolated transition response with a time shift can be expanded as a Taylor series,

$$s\left(t + j_k\right) = s(t) + \sum_{n=1}^{\infty} \frac{j_k^n}{n!} s^{(n)}(t), \tag{1.5}$$

where $s^{(n)}(t)$ is the n-th derivative of $s(t)$ at time t. Given that the position jitter in the system is small enough, $s(t)$ can be approximated very well by the first order expansion of its Taylor series,

$$s\left(t+j_k\right) \approx s(t)+j_k s'(t), \tag{1.6}$$

where

$$s'(t)=V_p \frac{\ln 3}{T_{50}}\left[\operatorname{sech}\left(\frac{\ln 3}{T_{50}}t\right)\right]^2. \tag{1.7}$$

Accordingly, the readback signal $r(t)$ becomes

$$r(t)=\sum_k b_k s\left(t-kT_b\right)+\sum_k b_k j_k s'\left(t-kT_b\right)+n(t). \tag{1.8}$$

The models for readback signals $r(t)$ in (1.4) and (1.8) are shown in Fig. 1.5.

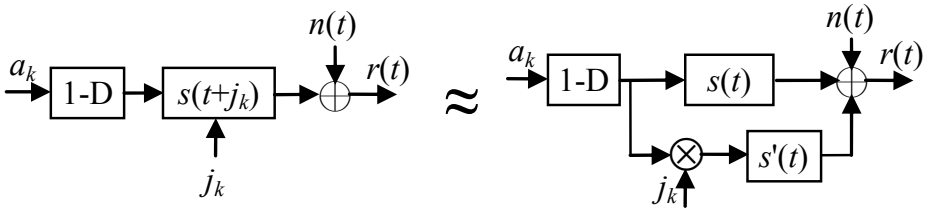
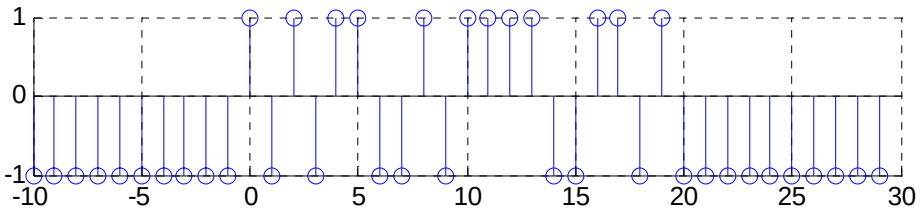


Fig. 1.5. The first order approximation of position jitter noise.

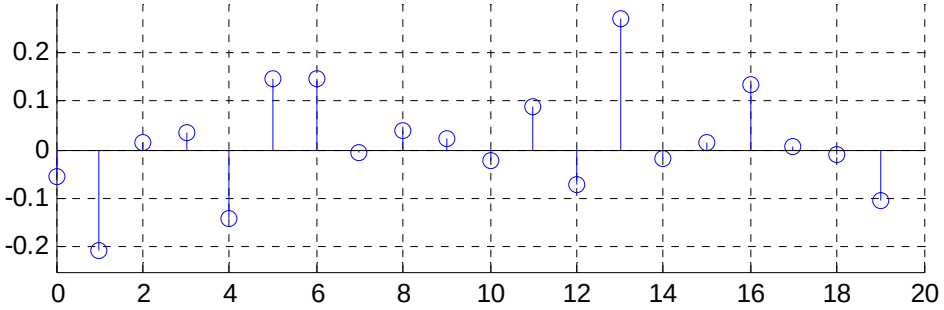
To demonstrate the effect of the position jitter noise, an example is given in Fig. 1.6, where the bipolar input sequence a_k, randomly generated jitter j_k and the continuous readback signals at D_c = 1 are presented. Note that the electronic noise $n(t)$ is not included. By comparing the waveforms in Fig. 1.6 (c), it is easy to see that large distortions happen with the occurrence of transitions in the input sequence.

1.1.3 Sampling the readback signal

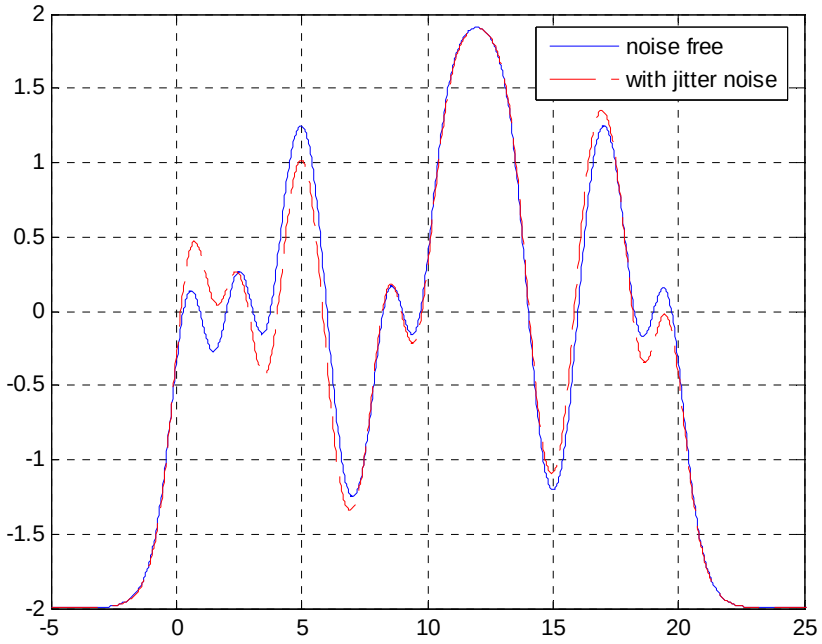
The readback signal of a perpendicular magnetic recording channel (PMRC) is a continuous time waveform. It will be much easier to convert it into a discrete time signal and then deal with it with highly developed digital signal processing techniques.



(a) Input sequence a_k



(b) Randomly generated position jitter input j_k



(c) Readback signals $r(t)$

Fig. 1.6. An example of readback signals $r(t)$, at $D_c = 1$.

Since the electronic noise $n(t)$ is assumed to have infinite bandwidth, before the sampling, a low pass filter is necessary to filter the readback signal and make sure the sampler will generate discrete signals with finite power. In this dissertation, we choose a matched filter $h(-t) = s(-t) - s(-t - T_b)$ and sample the continuous time signal at bit intervals (symbol rate). If the channel has only electronic noise, it is well known that the sampled signal is statistically sufficient with respect to the channel input sequence x_n. However, in the presence of position jitter noise, the statistical sufficiency may not be valid any more, which will make the subsequent channel detection suboptimal. But we still use this matched filter and the symbol rate sampler, since this configuration has low complexity and is still the one often used in current research on PMRCs.

1.1.4 Channel equalization

As shown in Fig. 1.4, the read channel for PMR is characterized by severe ISI, especially at high recording densities. Consequently, sampled signals are also corrupted by fairly long ISI. To mitigate the impairment of ISI, a digital filter, called an equalizer, is usually employed to equalize the channel response to some short partial-response target. In the past, commonly used partial-response (PR) targets were of the form $(1 - D)(1+D)^m$, such as PR4 ($1\text{-}D^2$), EPR4 ($1+D-D^2-D^3$), E^{2}PR4 ($1+2D-2D^3-D^4$), ME^{2}PR4 ($5+4D\text{-}3D^2\text{-}4D^3\text{-}2D^4$), etc. [7]. However, PR target polynomials with integer coefficients do not have a perfect spectral match to the channel response, especially at high recording densities. Instead, a technique was proposed in [8] to design optimized non-integer PR targets, which are also called generalized PR (GPR) targets. It has been proved that the channels equalized with optimized GPR targets significantly outperform the channels equalized with the conventional integer PR targets. The application of the equalization

with GPR targets on PMRCs is considered in [9] and [10], while we also utilize this technique in this dissertation.

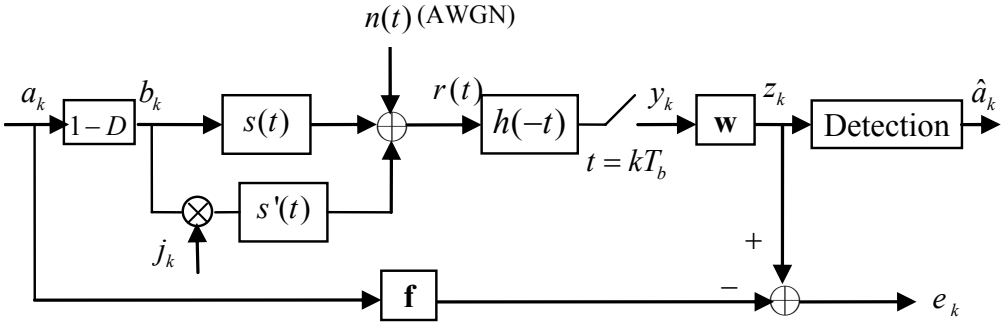


Fig. 1.7. Equalized PMRC with optimized GPR targets.

Shown in Fig. 1.7 is the equalized PMRC with optimized GPR targets, where the equalizer $\mathbf{w} = [w_{-N}, \ldots, w_0, \ldots, w_N]^{\mathrm{T}}$ is a $2N+1$ taps finite impulse response (FIR) digital filter, and $\mathbf{f} = [f_0, \ldots, f_{L_1-1}]^{\mathrm{T}}$ is the GPR target with length of L_1. Although the GPR target $\mathbf{f}$ could be anti-causal, we always use causal targets in this paper. Let $\mathbf{R}_{\mathbf{y}}$ be the $(2N+1)$-by-$(2N+1)$ autocorrelation matrix of y_k, with $\mathbf{R}_{\mathbf{y}}(i,j) = E\{y_{k-i}y_{k-j}\}$ for $-N \le i,j \le N$, $\mathbf{R}_{\mathbf{a}}$ the L_1-by-L_1 autocorrelation matrix of a_k, with $\mathbf{R}_{\mathbf{a}}(i,j) = E\{a_{k-i}\,a_{k-j}\}$ for $0 \le i,j \le L_1-1$, $\mathbf{R}_{\mathbf{y,a}}$ the $(2N+1)$-by-L_1 cross correlation matrix with $\mathbf{R}_{\mathbf{y,a}}(i,j) = E\{y_{k-i}\,a_{k-j}\}$ for $-N \le i \le N$ and $0 \le j \le L_1-1$. Then the mean-squared error can be expressed as

$$\mathrm{MSE} = E\left\{e_k^2\right\} = \mathbf{f}^{\mathrm{T}}\mathbf{R}_{\mathbf{a}}\mathbf{f} + \mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y,a}}\mathbf{f}\,. \tag{1.9}$$

By minimizing (1.9) and enforcing $f_0 = 1$, the optimized GPR target and equalizer can be computed by

$$\mathbf{f} = \lambda(\mathbf{R}_{\mathbf{a}} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R}_{\mathbf{y,a}})^{-1}\mathbf{C}\,, \tag{1.10}$$

$$\mathbf{w} = \mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R}_{\mathbf{y,a}}\mathbf{f}\,, \tag{1.11}$$

where $\mathbf{C} = [1, 0 \ldots 0]^T$ is a vector of length L_1, and

$$\lambda = \frac{1}{\mathbf{C}^T(\mathbf{R}_a - \mathbf{R}_{y,a}^T\mathbf{R}_y^{-1}\mathbf{R}_{y,a})^{-1}\mathbf{C}}.$$

1.1.5 Signal-to-noise ratio definition

In the evaluation of different signal processing and coding techniques for MRCs, we would like to compare the bit-error-rate (BER) or sector-error-rate (SER) performance with different error correcting codes and coding strategies on a given magnetic recording system, where the media/head pair and the associated electronic circuits are fixed. Since in magnetic recording, any code rate change leads to a change in recording density, an appropriate signal-to-noise ratio (SNR) definition should be independent of the recording density.

Traditionally, in a power-constraint additive white Gaussian noise (AWGN) channel, the SNR is defined as $R*E_b/N_0$, where R is the code rate, E_b is the average energy for each information bit, and N_0 is the single-sided noise spectral density height. This SNR definition is interpreted as a signal power to noise power ratio, by re-writing it as

$$\mathrm{SNR} = \frac{1}{2}\frac{R \cdot E_b/T_b}{N_0/2T_b} \tag{1.12}$$

where the factor 1/2 only causes a constant shift in logarithm domain, $10\log_{10}(\mathrm{SNR})$. In MRCs, the power-constraint is replaced with the transition response $s(t)$, which is fixed for a give magnetic recording system. Then for an MRC with only the electronic noise $n(t)$, we can define the SNR as

$$\mathrm{SNR} = \frac{E_i}{N_0} \tag{1.13}$$

where E_i is the average of each channel bit. However, since the energy of the transition

response $s(t)$ of a PMRC is infinite, it is not easy to define the E_i perfectly. In [11], E_i was a bit arbitrarily defined as

$$E_i = \int_{-\infty}^{\infty} \left(T_{50} s'(t)\right)^2 dt \,. \tag{1.14}$$

In [12], the integral is worked out to be

$$E_i = \frac{4 \ln 3}{3} T_{50} V_p^2 \,. \tag{1.15}$$

With a fixed transition response $s(t)$, the T_{50} and V_p are not varying with different recording densities; the E_i is also a constant in a given magnetic recording system. In other words, no matter how we define the E_i for a given magnetic recording system, distinct definitions only cause constant offsets in the logarithm domain. So a simpler definition of E_i is

$$E_i = T_{50} V_p^2 \,. \tag{1.16}$$

If we use T_{50} as the time unit for all signals, i.e., $T_{50} = 1$, then (1.16) becomes

$$E_i = V_p^2 \,, \tag{1.17}$$

which has a unit of energy and hence makes the SNR dimensionless.

With electronic noise only, the SNR definition in (1.13) is free of the recording density change, which is also wanted when we consider the jitter noise in the channel. For a MRC with electronic and jitter noise, we define the SNR as

$$\text{SNR} = \frac{E_i}{N_0 + M_0} \,, \tag{1.18}$$

where M_0 is the average transition noise energy associated with an isolated transition. Because we have only include position jitter noise (there is no pulse broadening noise), M_0 is actually the average position jitter noise energy associated with an isolated

transition. Note that we define M_0 as a kind of "energy" on purpose; we want M_0 to be a power spectral density just like N_0, i.e.,

$$\frac{M_0}{2T_b} = \frac{1}{T_b}\int_{-\infty}^{+\infty} \overline{n_j^2(t)}dt \,, \tag{1.19}$$

where n_j(t) denotes the jitter noise voltage waveform associated with each transition. By assuming the jitter noise is statistically independent among transitions, the right hand side of (1.19) is actually the in-band jitter noise power. Given that M_0 is considered a pseudo-power density height (single-sided) of the jitter noise, $M_0/(2T_b)$ is the integral of M_0 from 0 to $1/(2T_b)$. In [11], this property of M_0 is proved by simulations. That is, given a fixed variance of j_k, σ_j^2, the simulated in-band jitter noise power increases linearly with D_c, and M_0 could be the coefficient of D_c in this linear relationship.

From (1.19) we can derive several equivalent expressions of M_0. Let us list them below by the derivation order,

$$M_0 = 2\int_{-\infty}^{+\infty} \overline{n_j^2(t)}dt \,, \tag{1.20}$$

$$= 2\lim_{K\to\infty}\frac{1}{K}\int_{-\infty}^{+\infty}\left[\sum_{k=1}^{K} n_j(k,t)\right]^2 dt \,, \tag{1.21}$$

$$= 2E\left\{b_k^2\right\}\int_{-\infty}^{+\infty} E\left\{\left[s(t)-s(t+\Delta t_k)\right]^2\right\}dt \,, \tag{1.22}$$

$$= 2E\left\{b_k^2\right\}E\left\{\left\|s(t)-s(t+\Delta t_k)\right\|^2\right\} \,, \tag{1.23}$$

$$\approx 2E\left\{b_k^2\right\}\sigma_j^2\int_{-\infty}^{+\infty}\left(s'(t)\right)^2 dt \,, \tag{1.24}$$

where $n_j(k,t) = b_k\left[s(t-kT_b)-s(t-kT_b+j_k)\right]$ and (1.24) is obtained by using the first

order position jitter approximation. Since $a_k \in \{-1, 1\}$ and $b_k = a_k - a_{k-1}$, which gives $E\{b_k^2\} = 2$, we have

$$M_0 \approx 4\sigma_j^2 \int_{-\infty}^{+\infty} \left(s'(t)\right)^2 dt \,, \tag{1.25}$$

$$= \frac{16V_p^2 \ln 3}{3T_{50}} \sigma_j^2 \,, \tag{1.26}$$

where (1.26) is based on the integral result given in [12],

$$\int_{-\infty}^{+\infty} \left(s'(t)\right)^2 dt = \frac{4V_p^2 \ln 3}{3T_{50}} \,. \tag{1.27}$$

With this SNR definition, the jitter noise percentage is defined as $\alpha\% = \frac{M_0}{M_0 + N_0} \times 100\%$. Therefore, given a SNR and α%, we can compute the values of N_0 and M_0.

1.2 Bit-patterned magnetic recording channel

In high density BPM, the space between islands on both along-track direction and cross-track direction is very small. For example, given that the islands in BPM are squarely distributed, i.e., the bit period is equal to the track pitch, the track pitch should be about 25 nm to achieve an areal density of 1Tb/in^2, and about 18 nm to achieve an areal density of 2Tb/in^2. Due to the small track pitches, the read head flying above the media may not only sense the center track, which is the track under the center of the read head, but also the two tracks adjacent to the center track or even more tracks nearby. This phenomenon is called side reading or inter-track interference (ITI), which was usually ignored in PMR, where the track pitches are usually large enough. The ITI and ISI are

considered a two-dimensional (2D) interference in BPM, which needs to be modeled using the 2D response of an isolated island. In addition, there are specific noise sources in BPM, which make the modeling of BPMR channel more complicated than that of PMR.

In this section, we introduce the method to model the 2D response of BPM as well as the BPM-specific noises. Due to the strong ITI, the equalization of the BPMR channel presents a new challenge and we will not discuss it here, but we will investigate it in a separated chapter later in this dissertation.

1.2.1 Two-dimensional pulse response

Fig. 1.8 illustrates the three-dimensional (3D) geometry of a shielded MR (or GMR) read head and a patterned magnetic medium, where square islands and a soft under layer (SUL) are assumed in the medium. The MR (or GMR) element (sensor) is of length L, width W, and semi-infinite height, which has unit magnetic potential. The shields of zero magnetic potential are away from the MR (or GMR) element by G nanometers and are also semi-infinite on both along-track and cross-track direction. The read head is flying by d nanometers above the perpendicular magnetized island, which is of length a and thickness δ. The readback voltage is proportional to the signal flux injected into the MR (or GMR) element at the air-bearing surface (ABS), while the 2D signal flux can be modeled by a 3D evaluation of the reciprocity integral [13] [14],

$$\phi(x,z) = \frac{\mu_0}{i}\int_{-\infty}^{\infty} d\tilde{x}\int_{d}^{d+\delta} d\tilde{y}\int_{-\infty}^{\infty} d\tilde{z}\left[H_y(\tilde{x},\tilde{y},\tilde{z})M_y(\tilde{x}-x,\tilde{y},\tilde{z}-z)\right], \tag{1.28}$$

where μ_0 is the permeability of free space, i is the current in the imaginary coil, H_y is the read head magnetic field generated by the imaginary coil and M_y is the perpendicular magnetization of the medium.

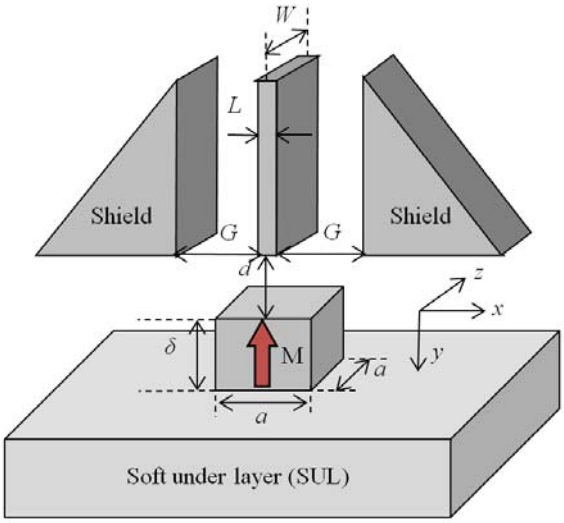


Fig. 1.8. Geometry of an MR/GMR read head and a patterned magnetic medium, where square islands and an SUL are assumed.

Since the magnetic field is the gradient of the magnetic potential, the reciprocity integral in (1.28) can be re-written as in [14]

$$\phi(x,z) = \frac{\mu_0}{i}\int_{-\infty}^{\infty} d\tilde{x}\int_{d}^{d+\delta} d\tilde{y}\int_{-\infty}^{\infty} d\tilde{z}\left[\psi(\tilde{x},\tilde{y},\tilde{z})\frac{\partial M_y(\tilde{x}-x,\tilde{y},\tilde{z}-z)}{\partial\tilde{y}}\right]. \tag{1.29}$$

Then to compute $\phi(x, z)$ we need to find the head magnetic potential function $\psi(x, y, z)$ for any point under the surface of the head, as well as the media magnetization M_y. To obtain $\psi(x, y, z)$, it is necessary to approximate the magnetic potential on the ABS first, which is $\psi_s(x, z) = \psi(x, 0, z)$. The $\psi(x, y, z)$ could be predicted in turn as a functional of $\psi_s(x, z)$ either in normal space [15] or in the Fourier transform domain [16]. Since the $\psi(x, y, z)$ is very important and interesting in the literature of the magnetic recording, people keep trying to get more accurate predictions for $\psi(x, y, z)$ [17]-[19].

On the other hand, the media magnetization is easy to handle, given some appropriate assumptions. Assuming that the island is uniformly magnetized, i.e., the perpendicular

magnetization $M_y(x, y, z)$ is a constant M in the range $d \le y \le d + \delta$, $-a/2 \le x, z \le a/2$, the derivative of M_y with respect to y turns out to be two impulse functions [20]. In addition, the effect of the magnetic SUL can be simplified by assuming that it is semi-infinite with infinite permeability. Then the head magnetic potential $\psi(x, y, z)$ in (1.29) need to include the magnetic potential of the image head, which is the mirror image of the real read head with respect to the boundary between the island and the SUL. So by considering a SUL, we can re-write (1.29) as [20]

$$\phi(x,z) = \frac{\mu_0}{i}\int_{-\infty}^{\infty} d\tilde{x}\int_{d}^{d+\delta} d\tilde{y}\int_{-\infty}^{\infty} d\tilde{z}\left\{\left[\psi(\tilde{x},\tilde{y},\tilde{z}) + \psi_{image}(\tilde{x},\tilde{y},\tilde{z})\right]\frac{\partial M_y(\tilde{x}-x,\tilde{y},\tilde{z}-z)}{\partial\tilde{y}}\right\}. \quad (1.30)$$

By taking our assumption on M_y, the reciprocity integral in (1.30) simplifies to

$$\phi(x,z) = C\int_{-\infty}^{\infty} d\tilde{x}\int_{-\infty}^{\infty} d\tilde{z}\left\{M(\tilde{x}-x,\tilde{z}-z)\left[\psi(\tilde{x},\tilde{y}=d,\tilde{z}) - \psi(\tilde{x},\tilde{y}=d+2\delta,\tilde{z})\right]\right\}, \quad (1.31)$$

where C is a constant and $M(x, z) = M$ for $-a/2 \le x, z \le a/2$, $M(x, z) = 0$ otherwise. Finally, the 2D readback voltage is computed as

$$V(x,z) = C_1\phi(x,z), \quad (1.32)$$

where C_1 is a constant. Since we always use the normalized readback voltage as the 2D pulse response of an isolated island, the constant C and C_1 do not matter.

In this dissertation, we will not make any contribution to improve the modeling of the BPM 2D response; we only use available or even simplified ones in our investigation of BPMR channels. But we would like to present a simple example to provide a clear understanding of the BPM 2D response modeling.

According to Fig. 1.8, we consider a medium-head pair with W = 15 nm, L = 4 nm, G = 6 nm, d = 10 nm, δ = 10 nm and a = 11 nm. As we discussed above, we need to start with a magnetic potential function on the ABS. In this example, we take the very simple

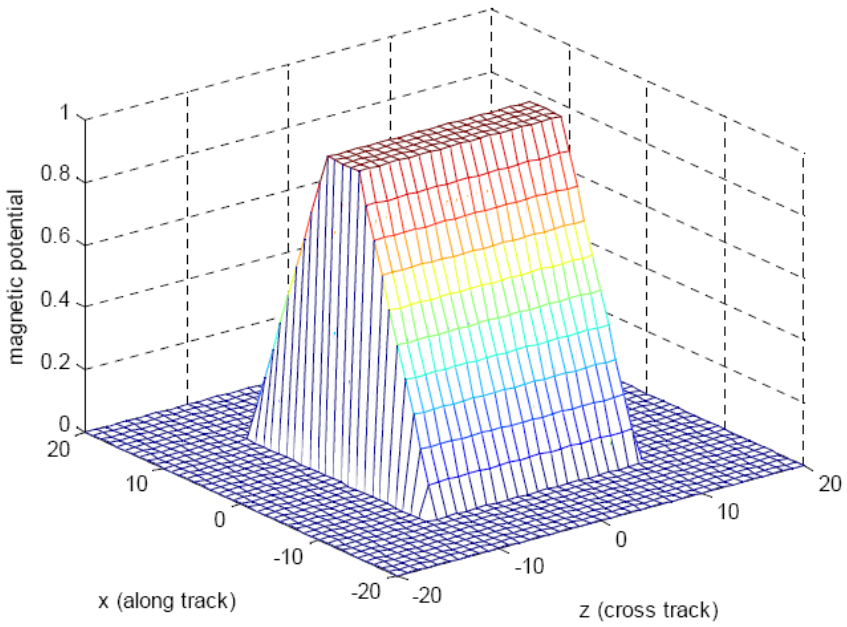
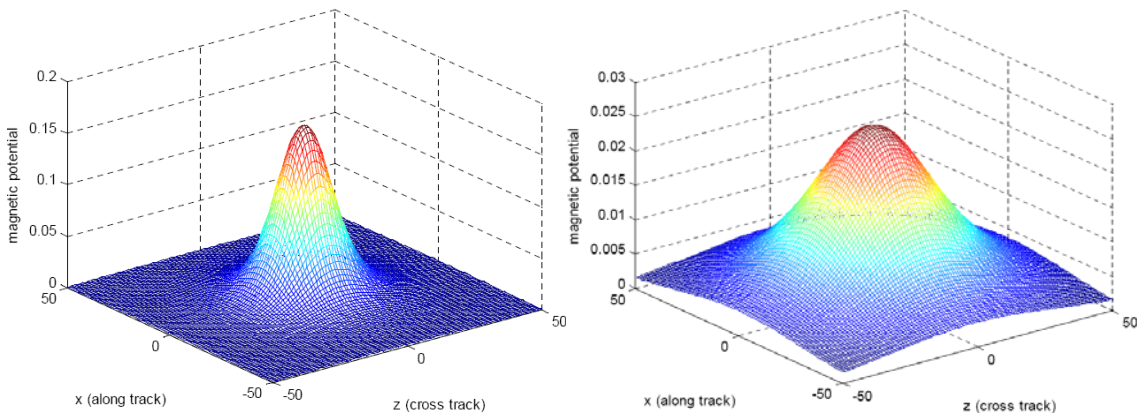


Fig. 1.9. $\psi_s(x, z)$, the magnetic potential of double-shielded read head on ABS by simple linear assumption.

one in [15], where potential distribution on the ABS is approximated by

$$\psi_s(x,z)=\begin{cases}\left(G+\frac{L}{2}+x\right)\Big/G & \text{for } -G-\frac{L}{2}\le x<-\frac{L}{2}, -\frac{W}{2}\le z\le\frac{W}{2};\\ 1 & \text{for } -\frac{L}{2}\le x<-\frac{L}{2}, -\frac{W}{2}\le z\le\frac{W}{2};\\ \left(G+\frac{L}{2}-x\right)\Big/G & \text{for } \frac{L}{2}<x\le G+\frac{L}{2}, -\frac{W}{2}\le z\le\frac{W}{2};\\ 0 & \text{elsewhere.}\end{cases} \tag{1.33}$$

This magnetic potential function is drawn in Fig. 1.9, where we can clearly see that the potential is assumed to be linearly attenuated between the MR element and the shields. Although it is specified in [15] that this read head is single-shielded (in two sides), the $\psi_s(x, z)$ in (1.33) tells us that the read head is actually double-shielded (on four sides) [21]. Given that $\psi(x, 0, z) = \psi_s(x, z)$, we can predict $\psi(x, y, z)$, the magnetic potential at any point under the read head by a functional [15]



(a) (b)

Fig. 1.10. (a) $\psi(x, 10, z)$, the magnetic potential on the plane of $y = d = 10$ nm. (b) $\psi(x, 30, z)$, the magnetic potential on the plane of $y = d + 2\delta = 30$ nm.

$$\Psi(x,y,z) = \frac{y}{2\pi}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{\psi_s(\tilde{x},\tilde{z})d\tilde{x}d\tilde{z}}{\left[(x-\tilde{x})^2, y^2, (z-\tilde{z})\right]^{3/2}}. \tag{1.34}$$

Fortunately, for the simple $\psi_s(x, z)$ in (1.33), the integral in (1.34) has a close form expression [22],

$$\begin{aligned}\psi(x,y,z) = &\left(\left\{\frac{y}{4\pi G}\log\left[\frac{R+(z-\tilde{z})}{R-(z-\tilde{z})}\right]+\frac{G+L/2+x}{2\pi G}\tan^{-1}\left[\frac{(z-\tilde{z})(x-\tilde{x})}{yR}\right]\right\}\Bigg|_{\tilde{x}=-G-L/2}^{-L/2}\right.\\ &+\frac{1}{2\pi}\tan^{-1}\left[\frac{(z-\tilde{z})(x-\tilde{x})}{yR}\right]\Bigg|_{\tilde{x}=-L/2}^{L/2}\\ &\left.+\left\{-\frac{y}{4\pi G}\log\left[\frac{R+(z-\tilde{z})}{R-(z-\tilde{z})}\right]+\frac{G+L/2-x}{2\pi G}\tan^{-1}\left[\frac{(z-\tilde{z})(x-\tilde{x})}{yR}\right]\right\}\Bigg|_{\tilde{x}=L/2}^{L/2+G}\right)\Bigg|_{\tilde{z}=-W/2}^{W/2},\end{aligned} \tag{1.35}$$

where $R = \sqrt{(x-\tilde{x})^2 + y^2 + (z-\tilde{z})^2}$. Then we draw $\psi(x, 10, z)$, the magnetic potential on the plane $y = d = 10$ nm in Fig. 1.10 (a) and $\psi(x, 30, z)$, the magnetic potential on the plane $y = d + 2\delta = 30$ nm in Fig. 1.10 (b). Finally, the normalized signal flux $\phi(x, z)$,

which is computed by the reciprocity integral in (1.31) is drawn in Fig. 1.11. To illustrate the ITI caused by high density BPM, we assume the islands are squarely distributed with the bit period and the track pitch of 18 nm and we draw the along-track pulse and the cross track profile as well as the pulse read back from the side tracks in Fig. 1.12.

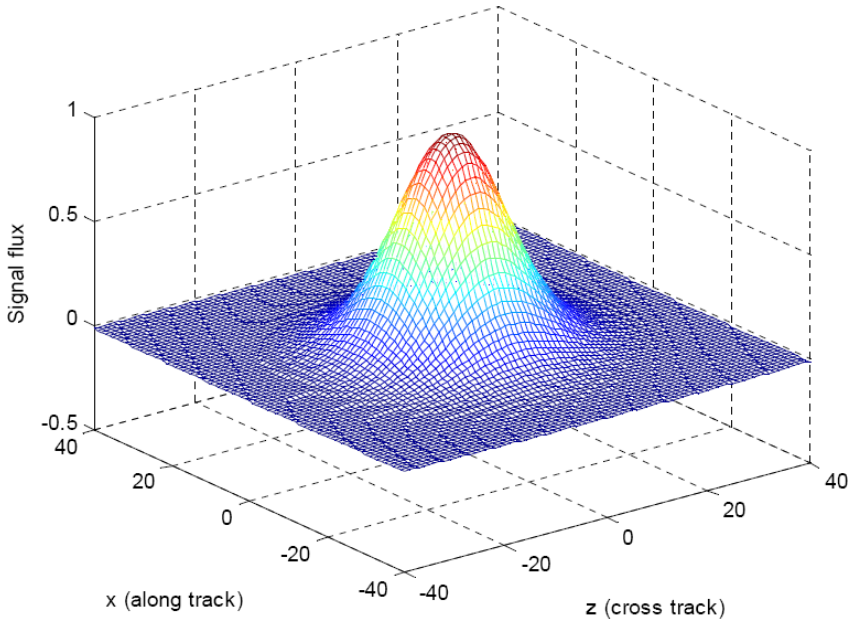


Fig. 1.11. The normalized signal flux $\phi(x, z)$.

1.2.2 Noises

Again, electronic noise (or thermal noise) is always considered a noise component in the read channel of any magnetic recording. But in high density BPMR channels, the there are several major noise sources which are BPM-specific and different from those in conventional magnetic recording systems. ITI (or side reading) is one of the BPM-specific noises which we have discussed earlier, while there are other two types of noises in BPMR channels: written-in errors and media noise.

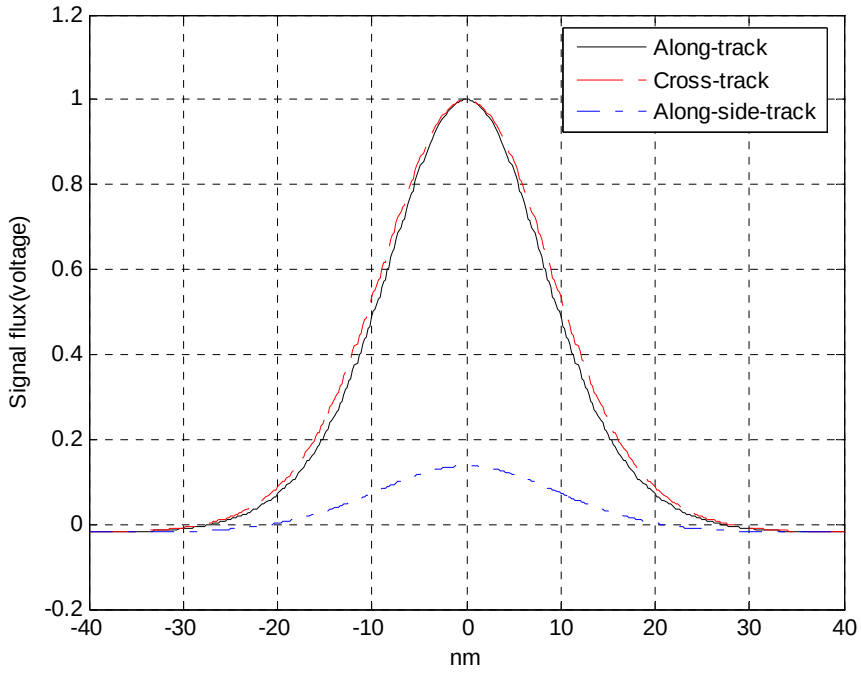


Fig. 1.12 The along-track pulses and the cross-track profile.

In BPMR, each magnetic island is used to store one bit and all islands are separated by non-magnetic material. So there is no transition noise and the 1 – D differentiation factor is gone. But during the write process, the write field needs to be carefully synchronized to make sure the write window is on the islands. However, in an actual medium, the islands may not be perfectly distributed and may not have uniform size and switching field. The island location fluctuations and the disturbing fields from neighbor islands may make an overwriting on one island and leave the island next to it unwritten [23], [24]. In addition, even if the time window for writing is perfectly on an island, the bit may not be successfully stored on the island when the switching field of the island is stronger than the head field. These errors occur during the write process and are called written-in errors. If we denote p as the probability of a bit after writing being different from the desired value, the written-in errors can be modeled by a binary symmetric channel [24].

Imperfect fabrication of the BPM causes the media noise in BPMR systems, where the major noise sources of the media noise are the fluctuations of island location, size, height, shape and saturated magnetization. The island location jitter can occur in both along-track and cross-track directions. The variation on island size may change the amplitude, the pulse widths of the 2D response on both along-track and cross-track directions. To simplify the modeling of the media noise, analytic functions of form $h(x, z) = h_x(x)h_z(z)$ could be used to approximate the actual 2D responses of the media, where $h_x(x)$ and $h_z(z)$ fit the along-track pulse and the cross-track profile, respectively. In [20], for a particular media-head pair, the 2D response is modeled by a 2D Gaussian pulse,

$$h(x,z) = h_x(x)h_z(z) = A\exp\left(-\frac{1}{2}\left(\frac{c^2x^2}{w_x^2}+\frac{c^2z^2}{w_z^2}\right)\right), \tag{1.36}$$

where A is the pulse amplitude, $w_x = \mathrm{PW}_{50\_\mathrm{along}}$ is the pulse width at half maximum on the along-track direction, $w_z = \mathrm{PW}_{50\_\mathrm{cross}}$ is the profile width at half maximum on the cross-track direction, $c = 2\sqrt{2\ln 2}$ is a constant used to associate the PW_{50} to the standard deviation of the Gaussian function. By assuming the pulse amplitude, w_x and w_z are linear functions of the island size, the 2D response taking into account the fluctuations of island size and location can be expressed as [20]

$$\tilde{h}(x,z) = (A+\Delta A)\exp\left(-\frac{1}{2}\left(\frac{c^2(x+\Delta x)^2}{(w_x+\Delta w_x)^2}+\frac{c^2(z+\Delta z)^2}{(w_z+\Delta w_z)^2}\right)\right), \tag{1.37}$$

which could be in turn approximated by the first order Taylor series expansion as

$$\tilde{h}(x,z) \approx h(x,z)+\left[\Delta x\frac{\partial h(x,z)}{\partial x}+\Delta z\frac{\partial h(x,z)}{\partial z}+\Delta w_x\frac{\partial h(x,z)}{\partial w_x}+\Delta w_z\frac{\partial h(x,z)}{\partial w_z}+\Delta A\frac{\partial h(x,z)}{\partial A}\right]. \tag{1.38}$$

Since the modeling and equalization are complicated and depend on various conditions, we will not continue our discussion in this section. Later in this dissertation, we will investigate equalization and detection methods for BPMR channels, where appropriate models of the read channel will be considered.

1.3 Magnetic recording system

For different kinds of MRCs, such as LMR, PMR or BPMR channels, the readback signal is corrupted by noises, which could be electronic noise, ISI, ITI and media noise. To recover the user data from the readback signal, the read channel is usually coded and additional components, such as timing recovery and gain control, may be needed in a practical implementation. In this dissertation, we are considering the magnetic recording system from a signal processing point of view and perfect timing and gain control are assumed. Shown in Fig. 1.13 is the model for a simple magnetic recording system.

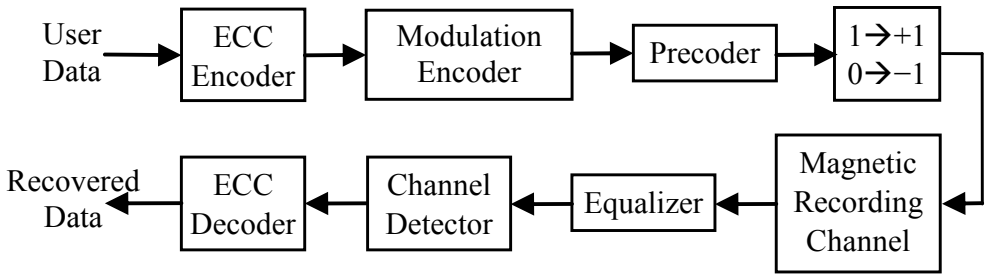


Fig. 1.13. A model of magnetic recording system

1.3.1 Channel detector

We have introduced equalized MRCs in the previous sections. After we get the signal at the output of the equalizer, a channel detector is responsible for estimating the bits recorded on the magnetic media. The channel detector is implemented by a detection

algorithm working on the trellis constructed according to the PR target, while the detection algorithm could output hard decisions or soft information for the recorded bits. With a channel detector which gives hard decisions of bits, such as the Viterbi algorithm (VA) [25], the outer error correcting code (ECC) such as the Reed-Solomon code [26] can only perform the hard decoding. With a channel detector which provides soft information for the bits, such as the soft output VA (SOVA) [27] or the Bahl-Cocke-Jelinek-Raviv (BCJR) algorithm [28], the outer ECC is capable of doing soft iterative decoding, which greatly enhances the BER and the SER performance of the magnetic recording system. Note that, for an ideal PR channel with AWGN, BCJR is considered a maximal *a posteriori* (MAP) detector, which is optimal in the sense of minimizing the probability of bit errors, while SOVA is a sub-optimal detector obtained as an extension of VA to provide soft output. For the perpendicular magnetic recording, the transition jitter noise is data dependent due to the differential factor 1 – D, which enables the use of the pattern-dependent noise predictive (DPNP) detector [29], [30], to further improve the detection and decoding performance.

1.3.2 Modulation code and precoder

At the input of the MRC, a modulation encoder is usually used to put some constraints on the input sequence for various purposes.

One purpose of the modulation codes is to facilitate the timing recovery and improve the distance properties of PRML channels, where two classes of modulation codes are often used, namely run length limited (RLL) codes [31] and maximum transition run (MTR) codes [32]. RLL codes are characterized by two parameters d, and k, which constrain the number of nontransitions between any two transitions to be at least d and at

most k. A (d, k) RLL code with $d > 0$ helps reducing the impact of the noise caused by consecutive transitions, while the maximum run length parameter k guarantees an adequate transition frequency for synchronization of the read clock. On the other hand, MTR codes limit the maximum run lengths of both transitions and nontransitions. The constraint on the maximum number of consecutive transitions eliminates certain minimum-distance error events, while the limitation on run length of nontransitions is assisting with timing recovery, just like the k parameter in RLL code.

Since the encoding of RLL codes and MTR codes are usually in non-return-to-zero-inverted (NRZI) space, where each 0's stand for nontransitions and 1's stand for transitions, it is necessary to use a precoder to convert the NRZI sequence in to non-return-to-zero (NRZ) format.

Another type of modulation codes shapes the channel input sequence into different distributions, where a typical example is the matched information rate (MIR) code proposed in [33]. The MIR code is a trellis code, which converts the independent uniformly distributed (i.u.d.) channel input into a sequence whose distribution mimics the distribution of the optimized Markov source, which achieves a higher information rate than the i.u.d. input on the PR channel. Since the MIR code encodes the input sequence in the NRZ space, no precoder is needed for this system.

1.3.3 Error correcting code

To overcome the noise and distortion in the read back signal and successfully recover the user data, ECCs are always considered in today's magnetic recording systems. Although ECCs could be nonlinear, we only consider linear codes in this dissertation. A (n, k) binary linear code encodes k-bit long information words into n-bit long codewords,

giving a code rate of $R = k/n$. During the investigation of different ECCs for magnetic recording systems, it is necessary to keep the same user density, $D_u = R * D_c$, which means that lower code rates correspond to higher recording densities. Since the increase of the recording density D_c boost up the ISI in MRCs, we need to keep the code rate above some level, which is typically 0.9.

The Reed-Solomon (RS) code is a widely used ECC in current magnetic recording systems, and it guarantees the correction of up to a certain number of symbol errors. Nowadays, soft iterative decoding techniques are being considered for magnetic recording, namely low-density parity-check (LDPC) codes [34] decoded by massage passing algorithms, which have been proved to significantly outperform the RS codes on various MRCs. The soft LDPC decoder can take more information from the channel detector than the hard RS decoder. Moreover, the soft output of the LDPC decoder can be fed back into the channel detector to implement the so called turbo equalization. This will refine the soft information for the bits and the BER and SER performance of the system is further improved.

1.4 Overview of the dissertation

In this dissertation we develop new channel detection and LDPC coding techniques for magnetic recording systems to improve their BER and SER performance. In Chapters 3−7, advanced channel detection, LDPC decoding algorithms and LDPC code design techniques are investigated for perpendicular magnetic recording systems, while a sophisticated channel detection method is proposed to mitigate the ITI in bit-patterned magnetic recording systems.

Before we present any contributions in this dissertation, the state-of-the-art channel detection and LDPC decoding techniques are described in Chapter 2, where we introduce the generic soft-input soft-output (SISO) channel detectors as well as the conventional message-passing decoding algorithms for binary and nonbinary LDPC codes,

In Chapter 3, improved channel detectors for nonbinary LDPC coded PR channels are investigated. We apply a sophisticated channel detector, namely the optimal subblock-by-subblock detector [48], to nonbinary LDPC coded PMRCs. Moreover, we derive a new symbol-based BCJR detector to do the turbo equalization accurately, since the one in [48] is working with unnecessary approximations when the turbo equalization is implemented.

In Chapter 4, an improved belief-propagation (BP) decoder is proposed for LDPC coded PR channels, where the new decoder takes into account the dependence between the channel messages produced by the channel detector. On LDPC coded PMRCs, the improved BP decoder provides significant gains over the standard BP decoder. Furthermore, this technique is extended to the decoding of nonbinary LDPC codes and additional gains are observed.

In Chapter 5, advanced LDPC code design techniques are investigated. Since short cycles on the factor graphs of LDPC code may severely degrade the decoding performance of LDPC codes, we are aiming at constructing LDPC codes with fewer short cycles to get LDPC codes with better performance, especially at high SNRs.

In Chapter 6, we investigate RS plus LDPC concatenated architectures for PMRC. By simulation, we find the optimal code rate and iterative scheme for the concatenated codes. The performance of the concatenated codes is compared in both random noise and media

defects. In addition, we estimate the performance of the concatenated codes whose inner LDPC codes have a column weight of two, at very high SNRs, where their error floors are found.

In Chapter 7, advanced channel equalization and detection methods for BPMR channels are investigated, and then a multi-track detection technique is proposed. This technique works with the equalizers which equalize the read channel to 2D GPR targets. Moreover, we find the performance bounds of the multi-track detection technique and develop multiple detection strategies to achieve the bounds on BPMR channels.

In Chapter 8, we give some conclusive discussions that can be drawn from our work and make some recommendations for future research.

2 Detection and Decoding of LDPC Coded PR Channels

Fig. 2.1 shows a PR channel model, where **f** is the response of the channel, μ_k is the channel input, y_k is the noisy channel output and the additive noise n_k is usually assumed to be AWGN. The input-output relation of the PR channel can be written as

$$y_k = \sum_i f_i \mu_{k-i} + n_k . \tag{2.1}$$

Since MRCs have long ISI, to mitigate the impairment of the ISI, the channels need to be equalized to short PR targets. The equalized MRCs are also PR channels, but the noise in the channels is not AWGN in general. It is easy to see that a_k, z_k and e_k in Fig. 1.7 correspond to μ_k, y_k and n_k in Fig. 2.1, respectively. Although the noise e_k in equalized MRCs may not be white and Gaussian, it could still be treated as AWGN during channel detection.

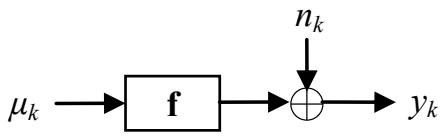


Fig. 2.1. A PR channel model.

In this dissertation, we are interested in LDPC coded MRCs with soft iterative decoding, which can be equivalently expressed by LDPC coded PR channels as in Fig. 2.2, where the channel detector has soft input and soft output, and the LDPC decoder is implemented by a message passing algorithm. In the initial run of the channel detection, there is no *a priori* information available for the channel bits. The channel detector works on the trellis constructed according to the response (PR target) **f** and produces the channel

messages (the *a posteriori* probabilities) for channel bits. The LDPC decoder takes the channel messages as input and tries to find a valid codeword by iterating on the message passing algorithm. If the message passing algorithm converges to a valid codeword within a predefined maximum number of iterations in the LDPC decoder, the decoding is considered successfully completed. Otherwise, the soft information generated by the LDPC decoder will be fed back to the input of the channel detector to do the channel iteration, which is also called turbo equalization. Note that, in general, the message passed between the channel detector and LDPC decoder is extrinsic information; the extrinsic information sent to the decoder (or detector) does not contain the information sent from that decoder (or detector). For example, the extrinsic information generated by the channel detector is computed by dividing the *a posteriori* probabilities (APPs) by the *a priori* probabilities, while the division becomes subtraction in the logarithm domain.

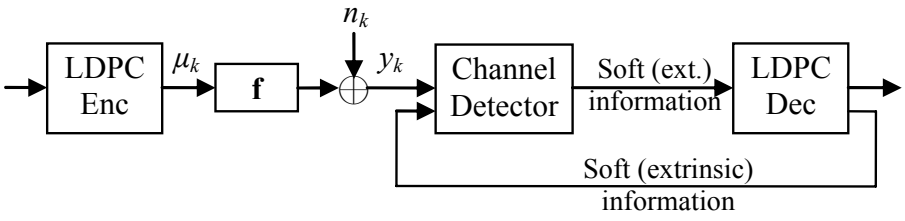


Fig. 2.2. A model for LDPC coded PR channel with soft iterative decoding.

In this chapter, we introduce the BCJR and SOVA algorithms, the two generic SISO channel detectors, as well as the BP algorithms for binary and nonbinary LDPC decoding and the min-sum algorithm which is a low complexity approximation of the BP algorithm.

2.1 SISO channel detectors

2.1.1 BCJR algorithm

The BCJR algorithm [28] is a MAP detector for PR channels with AWGN, which is optimal in the sense of minimizing the symbol error probability. Since we are considering PR channels with binary input, the BCJR detector is minimizing the bit probability.

Let the PR channel with v bits of memory be represented by a binary-input trellis without parallel transitions, with 2^v states at each stage. For an input block of N bits, $\boldsymbol{\mu}_1^N \triangleq (\mu_1, \mu_2, \cdots \mu_N)$, the noisy channel output is $\boldsymbol{y}_1^N \triangleq (y_1, y_2, \cdots y_N)$, and the corresponding state sequence of the channel is represented by a finite vector $\boldsymbol{s}_0^N \triangleq (s_0, s_1, \cdots, s_N)$, where the initial and final states, $s_0 = S_0$ and $s_N = S_N$, are known. The Markov model of the PR channel encoder can be represented as $P(s_k \mid \boldsymbol{s}_0^{k-1}, \boldsymbol{\mu}_1^k) = P(s_k \mid s_{k-1}, \mu_k)$. The APP sought is $P(\mu_k \mid y_1^N)$ with $1 \le k \le N$, which could be expanded as

$$\begin{aligned} P(\mu_k \mid \boldsymbol{y}_1^N) &= \frac{P(\mu_k, \boldsymbol{y}_1^N)}{P(\boldsymbol{y}_1^N)} \\ &= \frac{1}{P(\boldsymbol{y}_1^N)} \sum_{s_{k-1}} \sum_{s_k} P(s_{k-1}, s_k, \mu_k, \boldsymbol{y}_1^N) \\ &= \frac{1}{P(\boldsymbol{y}_1^N)} \sum_{s_{k-1}} P(s_{k-1}, s_k, \mu_k, \boldsymbol{y}_1^N), \end{aligned} \tag{2.2}$$

where, the double sum becomes a single sum, due to the fact that there is exactly one stopping state s_k, given the starting state s_{k-1} and the input symbol μ_k. In addition, we do not need to consider the term $P(\boldsymbol{y}_1^N)$ which is a constant once the block of signal is received. By some derivations, the joint probability in (2.2) can be decomposed as

$$P(s_{k-1}, s_k, \mu_k, \mathbf{y}_1^N) = P(s_{k-1}, \mathbf{y}_1^{k-1}) \cdot P(s_k, \mu_k, y_k \mid s_{k-1}) \cdot P(\mathbf{y}_{k+1}^N \mid s_k), \tag{2.3}$$

where the three terms are named the forward state probability, the branch transition probability and the backward state probability, and denoted by $\alpha_{k-1}(s_{k-1})$, $\gamma_k(s_{k-1},s_k)$ and $\beta_k(s_k)$, respectively. The forward and backward state probabilities can be computed recursively as

$$\alpha_k(s_k) \triangleq P(s_k, \mathbf{y}_1^k) = \sum_{s_{k-1}} \alpha_{k-1}(s_{k-1}) \gamma_k(s_{k-1}, s_k), \tag{2.4}$$

$$\beta_k(s_k) \triangleq P(\mathbf{y}_{k+1}^N \mid s_k) = \sum_{s_{k+1}} \beta_{k+1}(s_{k+1}) \gamma_{k+1}(s_k, s_{k+1}). \tag{2.5}$$

The branch transition probability is calculated by

$$\gamma_k(s_{k-1}, s_k) \triangleq P(s_k, \mu_k, y_k \mid s_{k-1}) = P(\mu_k) \cdot P(y_k \mid s_{k-1}, \mu_k, s_k), \tag{2.6}$$

where $P(\mu_k)$ is the *a priori* information of the bit and $P(y_k \mid s_{k-1}, \mu_k, s_k)$ is the channel transition probability, which can be computed according to the noise distribution.

Before we start the recursions to compute the forward and backward state probabilities, we initialize $\alpha_0(s_0 = S_0) = 1$ and $\alpha_0(s_0 \neq S_0) = 0$, $\beta_N(s_N = S_N) = 1$ and $\beta_N(s_N \neq S_N) = 0$. Usually, the BCJR algorithm is implemented in the logarithm domain, where the multiplications and additions in (2.2)−(2.6) become additions and log_sum_exp operations, $\log(e^a + e^b)$.

In practice, it is convenient for sending the messages by converting the APPs to log likelihood ratios (LLRs),

$$L(\mu_k) = \log \frac{P(\mu_k = 1 \mid y_1^N)}{P(\mu_k = -1 \mid y_1^N)}. \tag{2.7}$$

Then the extrinsic information is simply obtained by

$$L_e(\mu_k) = L(\mu_k) - L_a(\mu_k), \tag{2.8}$$

where

$$L_a(\mu_k) = \log \frac{P(\mu_k = 1)}{P(\mu_k = -1)} \tag{2.9}$$

is the LLRs of the *a priori* information.

2.1.2 SOVA

The BCJR algorithm is optimal for PR channels with AWGN, but its high computational complexity increases the cost of its implementation in commercial products. SOVA [27] is a low complexity but sub-optimal channel detection algorithm, which is obtained by simply adding reliability computations into the original VA. Therefore, let us start from the VA and go to SOVA smoothly.

VA is a PRML detector, or by another name, the maximum likelihood sequence detector (MLSD). From the maximum likelihood term, it is easy to understand that VA finds

$$\begin{aligned} \hat{\boldsymbol{\mu}}_1^N &= \arg\max_{\boldsymbol{\mu}_1^N} \left\{ P\left(\boldsymbol{y}_1^N \mid \boldsymbol{\mu}_1^N \right) \right\} \\ &= \arg\max_{\boldsymbol{\mu}_1^N} \left\{ \prod_{k=1}^{N} P\left(y_k \mid \sum_i f_i \mu_{k-i} \right) \right\}. \end{aligned} \tag{2.10}$$

Given that the noise is AWGN with zero mean and a power of $N_0/2$, (2.10) can be re-written as

$$\hat{\boldsymbol{\mu}}_1^N = \arg\max_{\boldsymbol{\mu}_1^N} \left\{ (\pi N_0)^{\frac{N}{2}} \exp\left[-\sum_{k=1}^{N} \left(y_k - \sum_i f_i \mu_{k-i} \right)^2 \Big/ N_0 \right] \right\}. \tag{2.11}$$

Transforming (2.11) to the logarithm domain gives

$$\begin{aligned}\hat{\boldsymbol{\mu}}_1^N &= \arg\max_{\boldsymbol{\mu}_1^N}\left\{\frac{N}{2}\log(\pi N_0) - \sum_{k=1}^{N}\left(y_k - \sum_i f_i \mu_{k-i}\right)^2 \Big/ N_0\right\} \\ &= \arg\min_{\boldsymbol{\mu}_1^N}\left\{\sum_{k=1}^{N}\left(y_k - \sum_i f_i \mu_{k-i}\right)^2\right\},\end{aligned} \tag{2.12}$$

where it is clear that the VA finds the data sequence $\hat{\boldsymbol{\mu}}_1^N$ which has minimum Euclidean distance from the actual transmitted sequence.

To see how the VA works, let us define the branch metric first,

$$\lambda_k(s_{k-1}, s_k) = (y_k - x_k)^2 / N_0, \tag{2.13}$$

where $x_k = \sum_i f_i \mu_{k-i}$ is the branch value for a particular transition from s_{k-1} to s_k. Note that we keep the $1/N_0$ factor in (2.13), which is not useful for the VA but scales the LLRs computed by SOVA. During the detection on the trellis of a PR channel, each trellis state maintains a path of hard decisions and an accumulated path metric. A state s_k at time k has two incoming paths, and the path with the minimum path metric is selected to survive and the path metric is updated as

$$M_k(s_k) = \min_{s_{k-1}}\left\{M_{k-1}(s_{k-1}) + \lambda_k(s_{k-1}, s_k)\right\}. \tag{2.14}$$

The detection starts from a know state $s_0 = S_0$, and the path metric at time 0 is initialized as $M_0(s_0 = S_0) = 0$ and $M_0(s_0 \neq S_0) = -\infty$. The detection continues to the end of the block, where the end state is known as $s_N = S_N$, then the survivor path that ends at S_N is considered the maximum likelihood (ML) path and the hard decisions on that path are the $\hat{\boldsymbol{\mu}}_1^N$ in (2.10)−(2.12). In practice, we do not need to wait for the detection output after the end of the block is reached. Since all survivor paths at time k may merge at a particular state at some time $k - D$, the hard decision on $\hat{\mu}_{k-D}$ can be made once all survivor paths

merge or the delay D is set to a large enough value.

SOVA is simply doing some additional work on the VA to generate soft information for the bits. Again, there are two paths that end at a state s_k at time k, via two states $s_{k-1}^{(1)}$ and $s_{k-1}^{(2)}$ at time $k-1$. According to [35], given that p_c denotes the probability that the survivor path decision made on s_k at time k is correct, the LLR $\log[p_c/(1-p_c)]$ can be approximated by the difference metric

$$\Delta_k = \left| \left[M_{k-1}\left(s_{k-1}^{(1)}\right) + \lambda_k\left(s_{k-1}^{(1)}, s_k\right) \right] - \left[M_{k-1}\left(s_{k-1}^{(2)}\right) + \lambda_k\left(s_{k-1}^{(2)}, s_k\right) \right] \right|. \tag{2.15}$$

Once all survivor paths merge at time $k-D$, or for a large enough delay D, we can choose the path with the lowest metric on s_k as the pseudo-ML path. By tracing from the state s_k at time k, along the pseudo-ML path, back to time $k-D$, there are $D+1$ paths which did not survive and were discarded; there are also $D+1$ difference metrics Δ_i for $k-D \le i \le k$ computed. Then the LLR of μ_{k-D} is approximated by

$$L\left(\mu_{k-D}\right) \approx \hat{\mu}_{k-D} \cdot \min\left\{\Delta_{k-D}, \Delta_{k-D+1}, \cdots, \Delta_k\right\}, \tag{2.16}$$

where the minimum is taken only over the non-survivor paths which have different decisions at time $k-D$ from the $\hat{\mu}_{k-D}$ on the pseudo-ML path.

We have mentioned that SOVA is also a SISO channel detector as the BCJR algorithm. To take into account the *a priori* information in SOVA, the branch metric in (2.13) need to be redefined as

$$\lambda_k\left(s_{k-1}, s_k\right) = \left(y_k - x_k\right)^2 / N_0 - \log P(\mu_k). \tag{2.17}$$

This branch metric is equivalent to the one in the BCJR algorithm, (see (2.6)), which makes SOVA an approximation to the MAP detector. More clearly, it has been shown in [36] that SOVA is closely related to the log-max-MAP algorithm, which is another

approximation to the BCJR algorithm. In addition, to facilitate the use of LLRs, the metric in (2.17) can be equivalently computed as [35]

$$\lambda_k\left(s_{k-1}, s_k\right) = \left(y_k - x_k\right)^2 \Big/ N_0 - \frac{1}{2}\mu_k L_a(\mu_k). \tag{2.18}$$

2.2 Low-density parity-check codes

LDPC codes were initially invented by Gallager [34] in 1962 and the work of MacKay [37] made them widely available since 1995. Davey and MacKay [38], [39] further generalized binary LDPC codes to finite fields GF(q) where $q = 2^p$. The use of BP decoding on binary LDPC codes has been shown to provide excellent performance over a wide variety of channels. Furthermore, nonbinary LDPC codes have been shown to perform even better than binary LDPC codes, albeit with higher decoding complexity, which was successfully lowered to a more tractable level by the use of a fast Fourier transform (FFT) in the BP decoder [39]-[41]. Both binary and nonbinary LDPC codes have been applied to magnetic recording channels, but nonbinary LDPC codes provide larger coding gains [42].

2.2.1 Introduction to LDPC codes

An LDPC code is a linear block code defined by an M by N sparse parity-check matrix $\mathbf{H}$, with $M < N$. A vector of length N, $\mathbf{x} = [x_1, \ldots, x_N]^T$ is a valid codeword if only if

$$\mathbf{Hx} = \mathbf{0}. \tag{2.19}$$

If the rank of $\mathbf{H}$ is $L \leq M$, then the LDPC code encodes $K = N - L$ information symbols in each codeword; the null space of $\mathbf{H}$ gives a (N, K) LDPC code, with a code rate of $R = K/N$. The symbols in LDPC codewords may or may not be binary. Since we are always

dealing with binary-input PR channels, we are particularly interested in the LDPC codes with symbols over GF(2^p). When $p = 1$, it is a binary LDPC code and each symbol in **x** corresponds to a bit; when $p > 1$, it is a nonbinary LDPC code and each symbol in **x** corresponds to a vector of p bits.

To facilitate the illustration of the iterative decoding of LDPC codes, the parity-check matrix **H** is usually represented by a bipartite graph, which is called a factor graph [43] or a Tanner graph [44]. The factor graph of an M by N parity-check matrix **H** has M check nodes and N variable nodes, which correspond to the M rows and N columns in **H**, respectively. The i-th check node and the j-th variable node is connected by an edge on the graph if only if there is an nonzero element at the intersection of the i-th row and the j-th column in **H**. Shown in Fig. 2.3 are a 3 by 6 parity-check matrix and its factor graph, where the variable nodes are denoted by circles and the check nodes are denoted by squares.

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

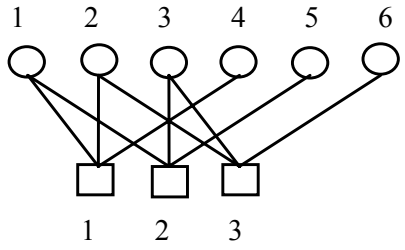


Fig. 2.3. Parity-check matrix (left) and its graph (right)

The number of edges connected to a node is defined as the degree of that node. The degree of a variable node is the number of the nonzero elements in its related column, while the degree of a check node is the number of the nonzero elements in its related row. If on a factor graph, all variable nodes have the same degree and all check nodes have

also the same degree (which may be different from the degree of the variable nodes), then the related code is a regular LDPC code. Otherwise, it is an irregular LDPC code.

2.2.2 Belief-propagation

BP is a message passing algorithm on factor graphs and is considered a very effective decoder for LDPC codes due to its low complexity (compared with the MAP decoder of linear codes) and good error rate performance.

The BP decoding of LDPC codes is executed iteratively, where each iteration consists of two steps: the checks-to-variables step (or the row step) and the variables-to-checks step (or the column step). Let $q^a_{n\to m}$ be the probability of $x_n = a$, sent from variable node n to check node m and $r^a_{m\to n}$ be the probability of $x_n = a$, sent from check node m to variable node n. $N(i)$ denotes the neighbors of node i, i.e., the set of nodes directly connected to node i on the factor graph. If i is a variable node then $N(i)$ is a set of check nodes; if i is a check node then $N(i)$ is a set of variable nodes.

At the beginning of the BP algorithm, the probability $q^a_{n\to m}$ is initialized by p^a_n, which is the local evidence of the variable node n. In the LDPC coded PR channel, p^a_n is a probability generated by the channel detector and used as the *a priori* information at the input of the BP decoder.

After the initialization, each check node m collects information from all of its neighbors and sends a message to each of them; this is the so called checks-to-variables step. Assuming that that variable node n is a neighbor of the check node m, then the message sent from check node m to variable node n can be computed by

$$r_{m\to n}^{a} = \sum_{\{x_j : j\in N(m)\setminus n\}} P\left(\{x_j\} \mid x_n = a, \mathbf{H}\right) \prod_{j\in N(m)\setminus n} q_{j\to m}^{x_j}, \tag{2.20}$$

where $N(m)\setminus n$ includes all of the neighbors of node m except node n, and $P(\{x_j\} \mid x_n = a, \mathbf{H})$ is a indicator function, which is equal to one if the values of $\{x_j\}$ satisfy the parity-checks in $\mathbf{H}$, given $x_n = a$; otherwise, it is equal to zero.

Next, in the variables-to-checks step, each variable node collects information from all check nodes connected to it and sends a message to each of them. The message sent from variable node n to check node m can be computed by

$$q_{n\to m}^{a} = \alpha_{nm} p_n^a \prod_{i\in N(n)\setminus m} r_{i\to n}^{a}, \tag{2.21}$$

where α_{nm} is a normalization factor.

Before the BP decoding goes back to the checks-to-variables step to start the next iteration, it is necessary to generate APPs for variable nodes to see if the decoding has been successful. The APP of variable node n is computed by

$$q_n^a = \alpha_n p_n^a \prod_{i\in N(n)} r_{i\to n}^{a}, \tag{2.22}$$

where α_n is also a normalization factor. Then the hard decisions are made on variable nodes according to their APPs. If all of the parity-checks in $\mathbf{H}$ are satisfied, then the decoding is successfully completed, and a valid codeword is found. Otherwise, the BP decoding continues to the next iteration. Usually we set a maximum number of BP iterations to prevent the algorithm from running forever.

2.2.3 Log-BP for binary LDPC codes

For binary LDPC code, the code symbols can only be zero or one; it is convenient to express their distributions by LLRs, where multiplications in (2.21) and (2.22) become

additions and the normalization factors are not needed. First, let us define the LLRs as $L(p_n)=\log\left(p_n^0/p_n^1\right)$, $L(r_{m\to n})=\log\left(r_{m\to n}^0/r_{m\to n}^1\right)$, $L(q_{n\to m})=\log\left(q_{n\to m}^0/q_{n\to m}^1\right)$ and $L(q_n)=\log\left(q_n^0/q_n^1\right)$. Then computations of the messages in (2.20) − (2.22) turn out to be

$$L(r_{m\to n})=2\tanh^{-1}\left(\prod_{j\in N(m)\backslash n}\tanh\left(L(q_{j\to m})/2\right)\right), \tag{2.23}$$

$$L(q_{n\to m})=L(p_n)+\sum_{i\in N(n)\backslash m}L(r_{i\to n}), \tag{2.24}$$

$$L(q_n)=L(p_n)+\sum_{i\in N(n)}L(r_{i\to n}). \tag{2.25}$$

2.2.4 Min-sum decoding

Although the implementation of the BP algorithm in the logarithm domain reduce the memory requirements and eliminate the normalization step, the evaluation of the hyperbolic tangent function is still of high computational complexity. To avoid the use of the hyperbolic tangent function and reduce the complexity as much as possible, an approximation can be made on the computation of the checks-to-variables messages,

$$L(r_{m\to n})\approx\left(\prod_{j\in N(m)\backslash n}\operatorname{sign}\left(L(q_{j\to m})\right)\right)\times\min_{j\in N(m)\backslash n}\left|L(q_{j\to m})\right|. \tag{2.26}$$

The sub-optimal decoding algorithm formed by (2.24) − (2.26) is the well known min-sum (MS) algorithm [45]-[47]. The MS algorithm is an approximation of the BP algorithm with very low complexity, since the multiplications of the signs {-1, +1} have trivial complexity; the only non-trivial operations in (2.26) are comparisons.

It has been shown in [46] that the checks-to-variables LLR messages in the MS algorithm have the same sign as those in the BP algorithm, but with larger amplitude,

which makes the error rate performance of the MS algorithm inferior to that of the BP algorithm. Therefore, the error rate performance is ought to be improved if we can reduce the amplitude of the LLR messages calculated in (2.26). Intuitively, there are two ways to do this task, as shown below.

$$L\left(r_{m\to n}\right)\approx\left(\prod_{j\in N(m)\backslash n}\operatorname{sign}\left(L\left(q_{j\to m}\right)\right)\right)\times\alpha\min_{j\in N(m)\backslash n}\left|L\left(q_{j\to m}\right)\right|, \tag{2.27}$$

$$L\left(r_{m\to n}\right)\approx\left(\prod_{j\in N(m)\backslash n}\operatorname{sign}\left(L\left(q_{j\to m}\right)\right)\right)\times\max\left\{\min_{j\in N(m)\backslash n}\left|L\left(q_{j\to m}\right)\right|-\beta,0\right\}, \tag{2.28}$$

where $0 < \alpha < 1$ and $\beta > 0$. The MS algorithm using (2.27) is the normalized MS algorithm, while the MS algorithm employing (2.28) is the offset MS algorithm. Both algorithms with fixed α and β can get remarkable improvement on error rate performance.

2.2.5 FFT-BP for nonbinary LDPC codes

For the decoding of nonbinary LDPC codes, the simplified methods we introduced for binary LDPC codes do not apply. Since the major complexity of nonbinary LDPC decoding is from the checks-to-variables step in (2.20), it is necessary to implement this step in a smart way.

A forward and backward algorithm was proposed in [38] to compute the check-to-variable message with lower complexity. First, the forward and backward partial sums are define as

$$\sigma_{mn}=\sum_{j:j\in N(m),j\le n}h_{mj}x_j\,, \tag{2.29}$$

$$\rho_{mn}=\sum_{j:j\in N(m),j\ge n}h_{mj}x_j\,, \tag{2.30}$$

where h_{mj} is the nonzero element at the m-th row and the j-th column in **H**. Then the distributions of the forward partial sums can be computed by a forward recursion. Given that i and j are adjacent indeces in $N(m)$ and $j > i$, the distribution of σ_{mj} can be calculated by

$$P\left(\sigma_{mj}=a\right)=\sum_{\{b,c:h_{mj}c+b=a\}}P\left(\sigma_{mi}=b\right)q_{j\to m}^{c}\,, \tag{2.31}$$

where a, b, $c \in \mathrm{GF}(2^p)$. The distributions of the backward partial sums can be computed in a similar way. Then the check-to-variable message is computed by

$$r_{m\to n}^{a}=\sum_{\{b,c:b+c+h_{mn}a=0\}}P\left(\sigma_{m(n-1)}=b\right)P\left(\rho_{m(n+1)}=c\right). \tag{2.32}$$

In [42], this forward and backward algorithm has been connected to the BCJR algorithm, where check m is looked at as a trellis with 2^p states and radix-2^p, in which the forward and backward partial sums are considered trellis states.

In [40], Richardson and Urbanke point out that the forward and backward algorithm in (2.32) and (2.31) is doing convolutions of a number of distributions, and the convolutions can be done in the Fourier transform domain. More clearly, given the distributions pmf (Q_i) for some random variables Q_i over $\mathrm{GF}(2^p)$, the distribution of the sum of the random variables can be computed by

$$\mathrm{pmf}\left(\sum_i Q_i\right)=\mathrm{IFFT}\left(\prod_i \mathrm{FFT}\left(\mathrm{pmf}\left(Q_i\right)\right)\right). \tag{2.33}$$

Note that all random variables in the forward and backward algorithm are over $\mathrm{GF}(2^p)$. So the FFT is not a simple 2^p -point Fourier transform but a p-dimensional 2-point FFT [42]. In this dissertation, we always use FFT-BP to implement the decoder for nonbinary LDPC code, since FFT-BP is especially good for the decoding of high rate LDPC codes.

3 Improved Detectors for Nonbinary LDPC Coded PR Channels

A magnetic recording channel can be modeled as a binary-input PR channel. An LDPC coded magnetic recording channel can be soft iteratively decoded, given that the channel detector is an SISO detector. Depending on the trade-off between complexity and performance, a decision can be made on whether or not to feed the soft output of the LDPC decoder back to the channel detector as *a priori* information. If the channel detector performs multiple iterations, this soft-iterative decoding is known as a turbo equalization system.

Two commonly used SISO channel decoders are the BCJR algorithm and SOVA, which we have introduced in Chapter 2. The BCJR algorithm is an optimum symbol-by-symbol channel detection algorithm. By "symbol" here we refer to data symbol in the context of channel signaling, not the code symbol in the context of error correction coding. Applied to a binary-input PR channel, the BCJR algorithm becomes an optimum bit-by-bit channel detection algorithm, which minimizes the probability of bit error. In this situation, we refer to it as the bit-based BCJR algorithm. SOVA is a sub-optimum channel detector and also bit-based in the same scenario.

For a binary LDPC decoder, the input probability information is either bit probabilities or LLRs. Using the output of the bit-based BCJR algorithm as the input information for the binary LDPC decoder is exactly the correct way to do turbo decoding. However, a nonbinary LDPC decoder needs the probabilities of nonbinary symbols in the codeword as the soft input. Since the probabilities for each symbol are actually the joint probabilities of p consecutive bits, they are usually generated by multiplying the bit probabilities at the output of the BCJR algorithm [38], [42]. However, the symbol

probability generated by multiplication of bit probabilities is only an approximation, which cannot be supported by theory.

To get accurate probabilities for code symbols, a new channel detector need to be employed; we find that the optimal subblock-by-subblock detector (OBBD) proposed in [48] does just that. Indeed, Cheng *et al.* [49] noticed the same problem when they worked on soft-decision decoding of RS coded PR channels; they have already applied the OBBD in their systems to achieve some performance improvement. The OBBD generates the joint probabilities of the data symbols in each subblock. For channels with binary signaling, it produces the joint probabilities of consecutive bits, which happen to be the symbol probabilities needed in nonbinary soft-iterative decoding.

However, this approach does not completely solve the problem. The OBBD takes bit probabilities as *a priori* information. If turbo equalization is implemented, all symbol probabilities output by the nonbinary LDPC decoder should be converted into bit probabilities before they are fed back to the input of the channel detector. Obviously, this is another unnecessary approximation.

In this chapter, we apply the OBBD to nonbinary LDPC coded PMRCs to evaluate the gains over those with the standard BCJR algorithm. As we discussed, this architecture is appropriate without turbo equalization. Furthermore, we extend the BCJR algorithm, in a similar way as the OBBD, to obtain an optimal symbol-by-symbol channel detection algorithm whose *a priori* information input and *a posteriori* probability output are both symbol probability information, and hence allow us to do turbo equalization in an exact manner. In addition, the simplifications of the new algorithm on PR channels as well as their complexities will also be carefully investigated.

Note that, because we are focusing on binary-input PR channels, the term "symbol" usually refers to the code symbol and not the data symbol, unless otherwise stated.

3.1 Application of the OBBD to nonbinary LDPC coded PMRCs

In [48], Hoeher gives the derivation of the OBBD as well as its simplifications for PR channels. Before simulating the read channel, we would like to give the set of equations for the OBBD but skip the derivation details. For consistency, we will keep the same notation as in the introduction of the BCJR algorithm in Chapter 2, which are different from those in [48].

Let the PR channel with v bits of memory be represented by a binary-input trellis without parallel transitions, with 2^v states at each stage. For an input block of N bits, $\boldsymbol{\mu}_1^N \triangleq (\mu_1, \mu_2, \cdots \mu_N)$, the noisy channel output is $\boldsymbol{y}_1^N \triangleq (y_1, y_2, \cdots y_N)$, and the corresponding state sequence of the channel is represented by a finite vector $\boldsymbol{s}_0^N \triangleq (s_0, s_1, \cdots, s_N)$, where the initial and final states, $s_0 = S_0$ and $s_N = S_N$, are known. We use $\underline{\boldsymbol{\mu}} \triangleq \boldsymbol{\mu}_{k-p+1}^k$ to denote the code symbol input at time k-p+1, which is mapped to the binary input sequence from time k–p+1 to time k. Then the probability distribution of a the subblock $\underline{\boldsymbol{\mu}}$ is computed as

$$\begin{aligned} P(\underline{\boldsymbol{\mu}} \mid \boldsymbol{y}_1^N) &= \frac{1}{P(\boldsymbol{y}_1^N)} \sum_{s_{k-p}} P(s_{k-p}, \boldsymbol{y}_1^{k-p}) \cdot P(s_k, \underline{\boldsymbol{\mu}}, \boldsymbol{y}_{k-p+1}^k \mid s_{k-p}) \cdot P(\boldsymbol{y}_{k+1}^N \mid s_k) \\ &= \frac{1}{P(\boldsymbol{y}_1^N)} \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \cdot P(s_k, \underline{\boldsymbol{\mu}}, \boldsymbol{y}_{k-p+1}^k \mid s_{k-p}) \cdot \beta_k(s_k), \end{aligned} \tag{3.1}$$

where $\alpha_{k-p}(s_{k-p})$ and $\beta_k(s_k)$ are the forward and backward state probabilities, respectively, which are calculated in the same way as in the standard BCJR algorithm. For

PR channels with v bits of memory, (3.1) is further simplified for two distinct cases: $p>v$ and $p\leq v$.

For $p>v$,

$$P(\underline{\boldsymbol{\mu}} \mid \mathbf{y}_1^N) = \frac{1}{P(\mathbf{y}_1^N)} \alpha_{k-p+v}(s_{k-p+v}) \cdot P(\underline{\boldsymbol{\mu}}'', s_k, \mathbf{y}_{k-p+v+1}^k \mid s_{k-p+v}) \cdot \beta_k(s_k), \tag{3.2}$$

where $\underline{\boldsymbol{\mu}}'' \triangleq \boldsymbol{\mu}_{k-p+v+1}^k$.

For $p\leq v$,

$$P(\underline{\boldsymbol{\mu}} \mid \mathbf{y}_1^N) = \frac{1}{P(\mathbf{y}_1^N)} \sum_{\{s_k \mid \mathbf{s}_k^{(v-p+1\to v)} = \underline{\boldsymbol{\mu}}\}} \alpha_k(s_k) \beta_k(s_k), \tag{3.3}$$

where $\mathbf{s}_k^{(v-p+1\to v)}$ are the latest p bits in the trellis registers at time k. These simplified versions of the OBBD expedite the channel detection significantly.

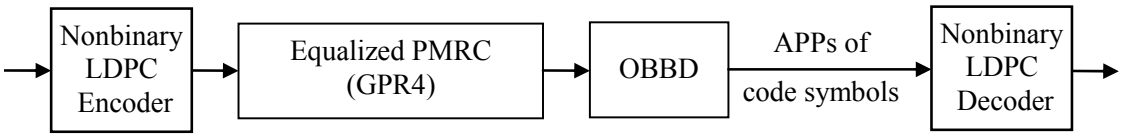


Fig. 3.1. A nonbinary LDPC coded PMR system using OBBD as the channel detector.

The system diagram of a nonbinary LDPC coded PMRC is shown in Fig. 3.1, where the turbo equalization is not implemented, since the OBBD is inappropriate for that, as we have discussed. We design a (911, 820) nonbinary LDPC code over GF(2^5) with rate 0.90011, using the progressive edge-growth (PEG) algorithm [50]. This LDPC code is approximately regular, with constant column weight three and row weights 30 and 31. In this system, the additive noise is 10% AWGN and 90% jitter noise power. PR targets are optimized at different working SNRs and have a fixed length of four. That means that there are always three bit registers in this PR channel, i.e., v = 3. In addition, in order to

highlight the performance difference between the OBBD and the standard BCJR algorithm, severe intersymbol interference is expected. Therefore, we simulate the read channel at a high recording density of 1.3596. The LDPC decoder performs at most 50 BP iterations. The simulation results are shown in Fig. 3.2, where the OBBD significantly improves the SER performance of the PMRC and achieves a SNR gain of more than 0.6 dB over the standard BCJR algorithm.

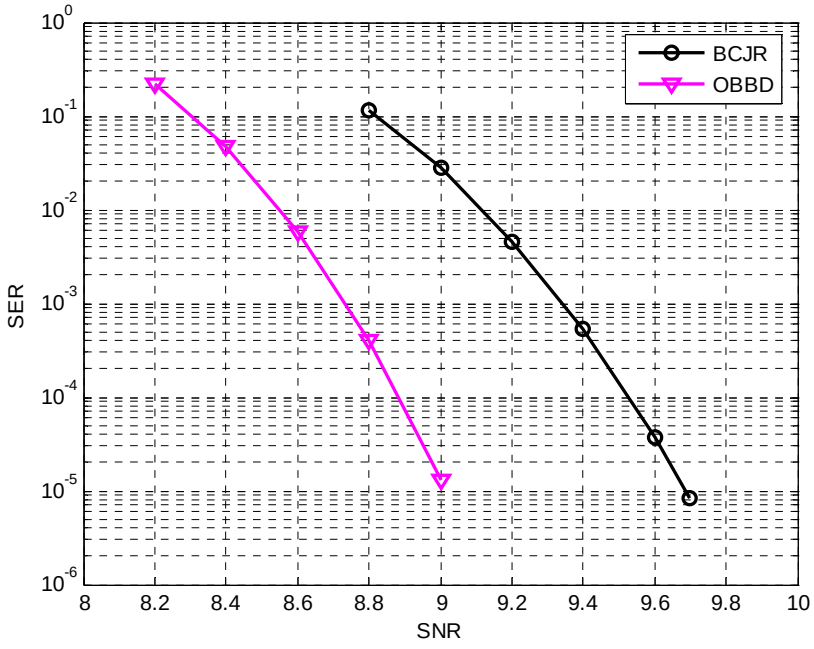


Fig. 3.2. Performance of nonbinary LDPC coded PMRCs with different channel detectors: BCJR and OBBD.

3.2 A symbol-based detection algorithm

Since the PMRCs using the OBBD cannot implement the turbo equalization exactly, it is necessary to design a new channel detector to do this job. In this section, we extend the BCJR algorithm, in a similar way as the OBBD, to derive a symbol-based channel

detection algorithm, which is optimal in the sense of minimizing the probability of code symbol errors for nonbinary coded PR channels.

We follow the consistent notation in this dissertation and give a detailed derivation of the proposed algorithm. One of the key points is that the *p* bits mapped into one symbol cannot be treated as independent variables, when we are trying to compute their joint probability. We can only assume that each symbol is independent from all others. Therefore, the Markov model of the channel encoder can be represented as $P(s_k \mid \mathbf{s}_0^{k-1}, \boldsymbol{\mu}_1^k) = P(s_k \mid s_{k-1}, \underline{\boldsymbol{\mu}})$, which is different from the model used in the original BCJR algorithm.

The symbol *a posteriori* probability sought is the conditional probability:

$$\begin{aligned} P(\underline{\boldsymbol{\mu}} \mid \mathbf{y}_1^N) &= \frac{P(\underline{\boldsymbol{\mu}}, \mathbf{y}_1^N)}{P(\mathbf{y}_1^N)} \\ &= \frac{1}{P(\mathbf{y}_1^N)} \sum_{s_{k-p}} \sum_{s_k} P(s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \mathbf{y}_1^N) \\ &= \frac{1}{P(\mathbf{y}_1^N)} \sum_{s_{k-p}} P(s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \mathbf{y}_1^N). \end{aligned} \tag{3.4}$$

In (3.4), the double sum becomes a single sum, because there is exactly one stopping state s_k, given the starting state s_{k-p} and the input symbol $\underline{\boldsymbol{\mu}}$. The summation over all starting states s_{k-p} is equivalent to the one over all the stopping states s_k. In addition, we do not need to consider the term $P(\mathbf{y}_1^N)$ which is common for different inputs $\underline{\boldsymbol{\mu}}$. The joint probability in (3.4) can be expressed as

$$\begin{aligned} &P(s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \mathbf{y}_1^N) \\ &\quad = P(s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^k, \mathbf{y}_{k+1}^N) \\ &\quad = P(\mathbf{y}_{k+1}^N \mid s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^k) \cdot P(s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^k) \end{aligned}$$

$$
\begin{aligned}
&= P(\boldsymbol{y}_{k+1}^{N} \mid s_k) \cdot P(s_k, \underline{\boldsymbol{\mu}}, \boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}, \boldsymbol{y}_1^{k-p}) \cdot P(s_{k-p}, \boldsymbol{y}_1^{k-p}) \\
&= P(\boldsymbol{y}_{k+1}^{N} \mid s_k) \cdot P(s_k, \underline{\boldsymbol{\mu}}, \boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}) \cdot P(s_{k-p}, \boldsymbol{y}_1^{k-p}) \\
&= \beta_k(s_k) \cdot \gamma_{(k-p+1,k)}^{\underline{\boldsymbol{\mu}}}(s_{k-p}, s_k) \cdot \alpha_{k-p}(s_{k-p}).
\end{aligned} \tag{3.5}
$$

where $\alpha_{k-p}(s_{k-p})$ is the forward state probability, $\beta_k(s_k)$ is the backward state probability, and the term $\gamma_{(k-p+1,k)}^{\underline{\boldsymbol{\mu}}}(s_{k-p}, s_k)$ is the branch transition probability associated with the branch from state s_{k-p} to state s_k with input $\underline{\boldsymbol{\mu}}$. Again, we must clarify the concept of the Markov model used in the above and following derivations; that is, events after a input symbol $\underline{\boldsymbol{\mu}}$ ($\underline{\boldsymbol{\mu}} \triangleq \boldsymbol{\mu}_{k-p+1}^{k}$) only depend on the stopping state s_k; we cannot say that events after a bit input at time *k* only depend on the stopping state s_k, unless it is the last bit in a symbol.

Let us see how to expand and calculate the forward and backward probabilities as well as the branch transition probability in this symbol-based algorithm. Starting with the forward probability

$$
\begin{aligned}
\alpha_k(s_k) &\triangleq P(s_k, \boldsymbol{y}_1^{k}) \\
&= \sum_{s_{k-p}} P(s_{k-p}, s_k, \boldsymbol{y}_1^{k}) \\
&= \sum_{s_{k-p}} P(s_{k-p}, \boldsymbol{y}_1^{k-p}) P(s_k, \boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}, \boldsymbol{y}_1^{k-p}) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) P(s_k, \boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \left(\sum_{\underline{\boldsymbol{\mu}}} P(\underline{\boldsymbol{\mu}}, s_k, \boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}) \right) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \left(\sum_{\underline{\boldsymbol{\mu}}} \gamma_{(k-p+1,k)}^{\underline{\boldsymbol{\mu}}}(s_{k-p}, s_k) \right).
\end{aligned} \tag{3.6}
$$

The calculation of the forward probability is recursive and its derivation is similar to that of the original BCJR algorithm. However, due to the symbol-based context, the second

summation term, which adds up the branch transition probabilities, is only over the symbols that connect the starting state s_{k-p} and the stopping state s_k. Similarly, we expand the backward probability as

$$\begin{aligned}
\beta_k(s_k) &\triangleq P(\boldsymbol{y}_{k+1}^N \mid s_k) \\
&= \sum_{s_{k+p}} P(s_{k+p}, \boldsymbol{y}_{k+1}^N \mid s_k) \\
&= \sum_{s_{k+p}} P(s_{k+p}, \boldsymbol{y}_{k+1}^{k+p}, \boldsymbol{y}_{k+p+1}^N \mid s_k) \\
&= \sum_{s_{k+p}} P(s_{k+p}, \boldsymbol{y}_{k+1}^{k+p} \mid s_k) P(\boldsymbol{y}_{k+p+1}^N \mid s_k, s_{k+p}, \boldsymbol{y}_{k+1}^{k+p}) \\
&= \sum_{s_{k+p}} P(s_{k+p}, \boldsymbol{y}_{k+1}^{k+p} \mid s_k) P(\boldsymbol{y}_{k+p+1}^N \mid s_{k+p}) \\
&= \sum_{s_{k+p}} \beta_{k+p}(s_{k+p}) P(s_{k+p}, \boldsymbol{y}_{k+1}^{k+p} \mid s_k) \\
&= \sum_{s_{k+p}} \beta_{k+p}(s_{k+p}) \left(\sum_{\underline{\boldsymbol{\mu}}} P(\underline{\boldsymbol{\mu}}, s_{k+p}, \boldsymbol{y}_{k+1}^{k+p} \mid s_k) \right) \\
&= \sum_{s_{k+p}} \beta_{k+p}(s_{k+p}) \left(\sum_{\underline{\boldsymbol{\mu}}} \gamma_{(k+1,k+p)}^{\underline{\boldsymbol{\mu}}}(s_k, s_{k+p}) \right).
\end{aligned} \tag{3.7}$$

The branch transition probabilities can be computed in the same way as in Cheng *et al.* [49],

$$\begin{aligned}
\gamma_{(k-p+1,k)}^{\underline{\boldsymbol{\mu}}}(s_{k-p}, s_k) &\triangleq P(s_k, \underline{\boldsymbol{\mu}}, \boldsymbol{y}_{k-p+1}^k \mid s_{k-p}) \\
&= P(\underline{\boldsymbol{\mu}} \mid s_{k-p}) \cdot P(s_k, \boldsymbol{y}_{k-p+1}^k \mid \underline{\boldsymbol{\mu}}, s_{k-p}) \\
&= P(\underline{\boldsymbol{\mu}}) \cdot P(s_k \mid \underline{\boldsymbol{\mu}}, s_{k-p}) \cdot P(\boldsymbol{y}_{k-p+1}^k \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k) \\
&= P(\underline{\boldsymbol{\mu}}) \cdot P(\boldsymbol{y}_{k-p+1}^k \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k).
\end{aligned} \tag{3.8}$$

The term $P(s_k \mid \underline{\boldsymbol{\mu}}, s_{k-p})$ is an indicator function, i.e.,

$$P(s_k \mid \underline{\boldsymbol{\mu}}, s_{k-p}) = \begin{cases} 1, & \text{if } s_{k-p} \text{ can reach } s_k \text{ with input } \underline{\boldsymbol{\mu}}\ ; \\ 0, & \text{otherwise.} \end{cases} \tag{3.9}$$

The term $P(\underline{\boldsymbol{\mu}})$ is the *a priori* probability of the symbol $\underline{\boldsymbol{\mu}}$, and the term $P(\boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k)$ is the channel transition probability. For PR channels with additive white noise, the channel transition probability of one symbol can be computed by multiplying the noise probabilities of all the bits. For the AWGN case, it becomes

$$\begin{aligned} P(\boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k) &= \prod_{0}^{p-1} P_n(y_{k-i} - x_{k-i}) \\ &= \left(\frac{1}{\sqrt{2\pi}\sigma} \right)^{p} \exp\left(\frac{-\sum_{0}^{p-1} \left(y_{k-i} - x_{k-i} \right)^2}{2\sigma^2} \right), \end{aligned} \tag{3.10}$$

where $(x_{k-p+1}, x_{k-p+2}, \cdots, x_k)$ are the noise free channel outputs corresponding to the input sequence $\underline{\boldsymbol{\mu}}$.

The derivation of the symbol-based detection algorithm is now completed. It is worth noting that for the first channel iteration, where the *a priori* probabilities are uniform, this symbol-based detection algorithm is equivalent to Hoeher's OBBD.

3.3 Simplified symbol-based detection for PR channels

A binary-input PR channel can be treated as a rate-one non-recursive convolutional encoder, whose states are defined by a subsequence of the input bit sequence. Let v represent the number of shift registers in this convolutional encoder. Two simplified versions of the symbol-based detection algorithm, which we derived in the previous section, are derived for the cases where $p>v$ and $p \leq v$.

3.3.1 Simplified algorithm for $p>v$

The summation in the last step in (3.4) can be re-written as

$$
\begin{aligned}
&\sum_{s_{k-p}} P(\underline{\mu}, s_{k-p}, s_k, \mathbf{y}_1^N) \\
&= P(\underline{\mu}, s_k, \mathbf{y}_1^N) \\
&= P(\underline{\mu}', \underline{\mu}'', s_k, \mathbf{y}_1^N) \\
&= P(\underline{\mu}'', s_{k-p+v}, s_k, \mathbf{y}_1^N) \\
&= P(\underline{\mu}'', s_{k-p+v}, s_k, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^{k-p+v}, \mathbf{y}_{k-p+v+1}^{k}, \mathbf{y}_{k+1}^{N}) \\
&= P(\mathbf{y}_{k+1}^N \mid \underline{\mu}'', s_{k-p+v}, s_k, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^{k-p+v}, \mathbf{y}_{k-p+v+1}^{k}) \\
&\quad \cdot P(\underline{\mu}'', s_{k-p+v}, s_k, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^{k-p+v}, \mathbf{y}_{k-p+v+1}^{k}) \\
&= P(\mathbf{y}_{k+1}^N \mid s_k) \cdot P(\underline{\mu}'', s_k, \mathbf{y}_{k-p+v+1}^{k} \mid s_{k-p+v}, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^{k-p+v}) \\
&\quad \cdot P(s_{k-p+v}, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^{k-p+v}) \\
&= P(\mathbf{y}_{k+1}^N \mid s_k) \cdot P(\underline{\mu}'', s_k, \mathbf{y}_{k-p+v+1}^{k} \mid s_{k-p+v}, \mathbf{y}_1^{k-p}, \mathbf{y}_{k-p+1}^{k-p+v}) \\
&\quad \cdot P(s_{k-p+v}, \mathbf{y}_1^{k-p+v}) \\
&= \beta_k(s_k) \cdot P(\underline{\mu}'', s_k, \mathbf{y}_{k-p+v+1}^{k} \mid s_{k-p+v}) \cdot \alpha_{k-p+v}(s_{k-p+v}),
\end{aligned}
\tag{3.11}
$$

where the input sequence $\underline{\mu}$ is divided into two parts $\underline{\mu}' \triangleq \mu_{k-p+1}^{k-p+v}$ and $\underline{\mu}'' \triangleq \mu_{k-p+v+1}^{k}$. Obviously $\underline{\mu}'$ is equivalent to the state at time *k-p+v*, i.e., s_{k-p+v}. The term $\beta_k(s_k)$ is the backward probability, which should be computed recursively using the method in Section 3.2. The other two terms can be expanded as follows:

$$
\begin{aligned}
&P(\underline{\mu}'', s_k, \mathbf{y}_{k-p+v+1}^{k} \mid s_{k-p+v}) \\
&= P(\underline{\mu}'' \mid s_{k-p+v}) \cdot P(s_k, \mathbf{y}_{k-p+v+1}^{k} \mid \underline{\mu}'', s_{k-p+v}) \\
&= P(\underline{\mu}'' \mid \underline{\mu}') \cdot P(s_k \mid \underline{\mu}'', s_{k-p+v}) \cdot P(y_{k-p+v+1}^{k} \mid s_{k-p+v}, \underline{\mu}'', s_k) \\
&= P(\underline{\mu}'' \mid \underline{\mu}') \cdot P(\mathbf{y}_{k-p+v+1}^{k} \mid s_{k-p+v}, \underline{\mu}'', s_k),
\end{aligned}
\tag{3.12}
$$

where $P(\underline{\mu}'' \mid \underline{\mu}')$ can be computed as

$$
P(\underline{\mu}'' \mid \underline{\mu}') = \frac{P(\underline{\mu}'', \underline{\mu}')}{P(\underline{\mu}')} = \frac{P(\underline{\mu})}{P(\underline{\mu}')}. \tag{3.13}
$$

The term $P(\boldsymbol{y}_{k-p+v+1}^{k} \mid s_{k-p+v}, \underline{\boldsymbol{\mu}}'', s_k)$ is the channel transition probability of $\underline{\boldsymbol{\mu}}''$, the subsequence of $\underline{\boldsymbol{\mu}}$. It can be calculated in the same way as $P(\boldsymbol{y}_{k-p+1}^{k} \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k)$ in Section 3.2. And

$$\begin{aligned}
\alpha_{k-p+v}(s_{k-p+v}) &\triangleq P(s_{k-p+v}, \boldsymbol{y}_1^{k-p+v}) \\
&= \sum_{s_{k-p}} P(s_{k-p}, s_{k-p+v}, \boldsymbol{y}_1^{k-p+v}) \\
&= \sum_{s_{k-p}} P(s_{k-p}, s_{k-p+v}, \boldsymbol{y}_1^{k-p}, \boldsymbol{y}_{k-p+1}^{k-p+v}) \\
&= \sum_{s_{k-p}} P(s_{k-p}, \boldsymbol{y}_1^{k-p}) \cdot P(s_{k-p+v}, \boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}, \boldsymbol{y}_1^{k-p}) \\
&= \sum_{s_{k-p}} P(s_{k-p}, \boldsymbol{y}_1^{k-p}) \cdot P(s_{k-p+v}, \boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \left(\sum_{\underline{\boldsymbol{\mu}}'} P(\underline{\boldsymbol{\mu}}', s_{k-p+v}, \boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}) \right) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \left(\sum_{\underline{\boldsymbol{\mu}}'} \gamma_{(k-p+1,k-p+v)}^{\underline{\boldsymbol{\mu}}'}(s_{k-p}, s_{k-p+v}) \right) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \cdot \gamma_{(k-p+1,k-p+v)}^{\underline{\boldsymbol{\mu}}'}(s_{k-p}, s_{k-p+v}),
\end{aligned} \tag{3.14}$$

where $\alpha_{k-p}(s_{k-p})$ is the forward state probability, which can be computed as in the original method in Section 3.2. The branch transition probability $\gamma_{(k+1,k+v)}^{\underline{\boldsymbol{\mu}}'}(s_k, s_{k+v})$ is similarly expressed as

$$\begin{aligned}
&\gamma_{(k-p+1,k-p+v)}^{\underline{\boldsymbol{\mu}}'}(s_{k-p}, s_{k-p+v}) \\
&\quad \triangleq P(s_{k-p+v}, \underline{\boldsymbol{\mu}}', \boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}) \\
&\quad = P(\underline{\boldsymbol{\mu}}' \mid s_{k-p}) \cdot P(s_{k-p+v}, \boldsymbol{y}_{k-p+1}^{k-p+v} \mid \underline{\boldsymbol{\mu}}', s_{k-p}) \\
&\quad = P(\underline{\boldsymbol{\mu}}') \cdot P(s_{k-p+v} \mid \underline{\boldsymbol{\mu}}', s_{k-p}) \cdot P(\boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}, \underline{\boldsymbol{\mu}}', s_{k-p+v}) \\
&\quad = P(\underline{\boldsymbol{\mu}}') \cdot P(\boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}, \underline{\boldsymbol{\mu}}', s_{k-p+v}).
\end{aligned} \tag{3.15}$$

Note that in the derivation of (3.14), we discarded the summation over $\underline{\boldsymbol{\mu}}'$ because $\underline{\boldsymbol{\mu}}'$ is equivalent to the state at time *k–p+v*, namely, s_{k-p+v}.

3.3.2 Simplified algorithm for $p \le v$

When the number of shift registers in the encoder is equal or larger than the bit length of a symbol, after each p-bit symbol is encoded, the p bits of the symbol are stored in the latest updated p bit registers. Fig. 3.3 depicts the encoder state at time k, where $s_k^{(i)}$ represents the i-th bit register of the state s_k.

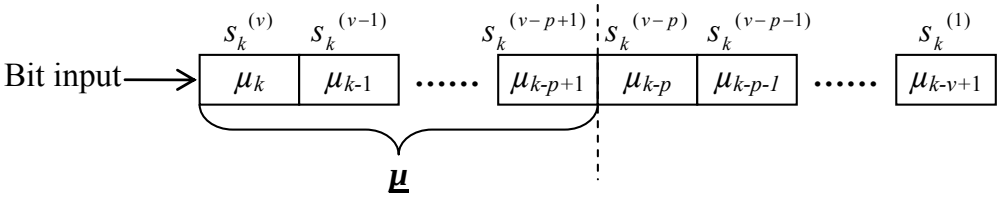


Fig. 3.3. The encoder state at time k, i.e., s_k. There are v bit registers $\mathbf{s}_k^{(1\to v)}$, in which $\mathbf{s}_k^{(v-p+1\to v)} = \underline{\boldsymbol{\mu}}$.

As in (3.4), the probability we want is

$$P(\underline{\boldsymbol{\mu}} \mid \mathbf{y}_1^N) = \frac{1}{P(\mathbf{y}_1^N)} P(\underline{\boldsymbol{\mu}}, \mathbf{y}_1^N).$$

We do not need to consider the term $P(\mathbf{y}_1^N)$ which is common for different inputs $\underline{\boldsymbol{u}}$. The joint probability $P(\underline{\boldsymbol{\mu}}, y_1^N)$ can be expanded as follows:

$$\begin{aligned} P(\underline{\boldsymbol{\mu}}, \mathbf{y}_1^N) &= P(\boldsymbol{\mu}_{k-p+1}^k, \mathbf{y}_1^N) \\ &= P(\textit{The latest p bits of } s_k = \boldsymbol{\mu}_{k-p+1}^k, \mathbf{y}_1^N) \\ &= P(\mathbf{s}_k^{(v-p+1\to v)} = \boldsymbol{\mu}_{k-p+1}^k, \mathbf{y}_1^N) \\ &= \sum_{\mathbf{s}_k^{(1\to v-p)}} P(\mathbf{s}_k^{(v-p+1\to v)} = \boldsymbol{\mu}_{k-p+1}^k, \mathbf{s}_k^{(1\to v-p)}, \mathbf{y}_1^N) \\ &= \sum_{\{s_k \mid \mathbf{s}_k^{(v-p+1\to v)} = \boldsymbol{\mu}_{k-p+1}^k\}} P(s_k, \mathbf{y}_1^N), \end{aligned} \tag{3.16}$$

where the joint probability is directly computed as

$$
\begin{aligned}
P(s_k, \mathbf{y}_1^N) &= \sum_{s_{k-p}} P(s_{k-p}, s_k, \mathbf{y}_1^N) \\
&= \sum_{s_{k-p}} \alpha_{k-p}(s_{k-p}) \gamma^{\underline{\boldsymbol{\mu}}}_{(k-p+1,k)}(s_{k-p}, s_k) \beta_k(s_k) \\
&= \alpha_k(s_k) \beta_k(s_k).
\end{aligned} \tag{3.17}
$$

The terms $\alpha_k(s_k)$ and $\beta_k(s_k)$ are the forward and backward probabilities in Section 3.2.

3.4 Complexity analysis

In this section, we discuss the implementation of three different versions of the algorithm and compare their complexity. The symbol-based detection algorithm and its two simplified versions need to compute the same forward and backward probabilities, given by (3.6) and (3.7) respectively. The algorithm steps are

1) Compute the channel transition probabilities $P(\mathbf{y}_{k-p+1}^k \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k)$ for all symbols in the block. For multiple turbo iterations, the channel transition probabilities are calculated only once.

2) Compute the branch transition probabilities $\gamma^{\underline{\boldsymbol{\mu}}}_{(k-p+1,k)}(s_{k-p}, s_k)$ for all symbols in the block. They are simply the multiplication of *a priori* probabilities $P(\underline{\boldsymbol{\mu}})$ and the channel transition probabilities $P(\mathbf{y}_{k-p+1}^k \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k)$.

3) Compute the forward and backward probabilities $\alpha_k(s_k)$ and $\beta_k(s_k)$ using the branch transition probabilities $\gamma^{\underline{\boldsymbol{\mu}}}_{(k-p+1,k)}(s_{k-p}, s_k)$.

After the computation of the forward and backward probabilities, the symbol-based detection algorithm and its two simplified versions use different ways to calculate the *a posteriori* probability $P(\underline{\boldsymbol{\mu}} \mid \mathbf{y}_1^N)$. Therefore, for different versions of the algorithm, we will

focus on the complexity analysis of the steps following the computation of the forward and backward probabilities.

As the bit-based BCJR, the symbol-based algorithm can be implemented in the log domain, where multiplications become additions, divisions become subtractions and additions turn into log sum operations, which can be implemented by table lookup. The lookup table we are using is the one in [7], where eight positive real numbers divide all positive real numbers into nine groups. The steps of a table lookup are:

1) Compare the two input values.
2) Get the positive difference by subtracting the smaller number from the larger number.
3) Fit the difference value into one of the nine real number groups.
4) If the difference value was not fitted into the group containing the largest real numbers, the table lookup result is the summation of the larger input number and a predefined real number assigned to the particular group. Otherwise, the table lookup result is the larger input number.

Assuming that the difference value would fall into nine groups with the same probability, a binary search could be a good choice for implementation of Step 3. Then Step 3 could be completed by three comparisons with probability of 7/9 or by four comparisons with probability of 2/9. To keep the analysis manageable, we approximate the complexity of Step 3 as three comparisons. Step 4 involves an addition with probability of 8/9; we approximate the complexity of Step 4 as one addition. Therefore, one table lookup operation takes about four comparisons, one addition and one subtraction. Addition and subtraction have a similar time complexity. Although a real number comparison is usually

a little bit slower than an addition, we can reasonably assume that they have similar time complexity, especially because the eight numbers in the log sum table have only four decimal digits, which makes the comparison faster. (We verified this assumption by testing the operation speed on various computers with different processors.) In this section, we measure the time complexity of one table lookup operation as x additions, and $x = 6$ is a good approximation.

3.4.1 Complexity of the symbol-based detection

According to (3.4) and (3.5), the symbol-based detection algorithm computes the joint probability

$$\begin{aligned} P(\underline{\boldsymbol{\mu}}, \boldsymbol{y}_1^N) &= \sum_{s_{k-p}} P(s_{k-p}, s_k, \underline{\boldsymbol{\mu}}, \boldsymbol{y}_1^N) \\ &= \sum_{s_{k-p}} \beta_k(s_k) \cdot \gamma_{(k-p+1,k)}^{\underline{\boldsymbol{\mu}}}(s_{k-p}, s_k) \cdot \alpha_{k-p}(s_{k-p}). \end{aligned} \tag{3.18}$$

The encoder has 2^v states. At each time k, $\underline{\boldsymbol{\mu}}$ could be 2^p possible symbols. For each possible symbol, it takes 2^{v+1} additions and $2^v - 1$ table lookup operations to compute the joint probability in (3.18).

3.4.2 Complexity of the simplified algorithm for $p > v$

This version of the simplified algorithm involves more steps than the original symbol-based algorithm. The following new values must be computed in order:

1) $P(\underline{\boldsymbol{\mu}}')$

2) $P(\boldsymbol{y}_{k-p+1}^{k-p+v} \mid s_{k-p}, \underline{\boldsymbol{\mu}}', s_{k-p+v})$

3) $\gamma_{(k-p+1,k-p+v)}^{\underline{\boldsymbol{\mu}}'}(s_{k-p}, s_{k-p+v})$ using (3.15)

4) $\alpha_{k-p+v}(s_{k-p+v})$ using (3.14)

5) $P(\underline{\mu}''|\underline{\mu}')$ using (3.13)

6) $P(\mathbf{y}_{k-p+v+1}^{k}|s_{k-p+v},\underline{\mu}'',s_k)$

7) $P(\underline{\mu}'',s_k,\mathbf{y}_{k-p+v+1}^{k}|s_{k-p+v})$ using (3.12)

8) $P(\underline{\mu},\mathbf{y}_1^N)=\sum_{s_{k-p}}P(\underline{\mu},s_{k-p},s_k,\mathbf{y}_1^N)$ using (3.11).

For binary-input PR channels with AWGN, the channel transition probabilities $P(\mathbf{y}_{k-p+1}^{k-p+v}|s_{k-p},\underline{\mu}',s_{k-p+v})$ and $P(\mathbf{y}_{k-p+v+1}^{k}|s_{k-p+v},\underline{\mu}'',s_k)$ can be directly obtained during the computation of $P(\mathbf{y}_{k-p+1}^{k}|s_{k-p},\underline{\mu},s_k)$ and do not take any additional operations. The marginal probabilities $P(\underline{\mu}')$ are calculated from the *a priori* probabilities $P(\underline{\mu})$, and their computation takes 2^{p-v}-1 table lookup operations for each possible sequence of $\underline{\mu}'$. For all 2^p possible symbols of $\underline{\mu}$, there are totally $2^v(2^{p-v}-1)$ table lookup operations needed.

The computation of $\gamma_{(k-p+1,k-p+v)}^{\underline{\mu}'}(s_{k-p},s_{k-p+v})$ takes only one more addition from $P(\mathbf{y}_{k-p+1}^{k-p+v}|s_{k-p},\underline{\mu}',s_{k-p+v})$ and $P(\underline{\mu}')$. For all 2^p possible symbols of $\underline{\mu}$, $\gamma_{(k-p+1,k-p+v)}^{\underline{\mu}'}(s_{k-p},s_{k-p+v})$ has to be calculated for 2^v possible sequence of $\underline{\mu}'$ and 2^v states of s_{k-p}, totally 2^{2v} additions.

Then, according to (3.14), the computational complexity of $\alpha_{k-p+v}(s_{k-p+v})$ for all 2^v states is 2^{2v} additions and $2^v(2^v-1)$ table lookup operations.

For each possible symbol of $\underline{\mu}$, the computation of $P(\underline{\mu}''|\underline{\mu}')$ requires one

operation in addition to $P(\underline{\mu}')$ and $P(\underline{\mu})$; $P(\underline{\mu}'', s_k, \mathbf{y}_{k-p+v+1}^k \mid s_{k-p+v})$ can be obtained with one additional operation from $P(\underline{\mu}'' \mid \underline{\mu}')$ and $P(\mathbf{y}_{k-p+v+1}^k \mid s_{k-p+v}, \underline{\mu}'', s_k)$. With two more operations, $P(\underline{\mu}, \mathbf{y}_1^N)$ can be calculated from $P(\underline{\mu}'', s_k, \mathbf{y}_{k-p+v+1}^k \mid s_{k-p+v})$, $\beta_k(s_k)$ and $\alpha_{k-p+v}(s_{k-p+v})$. Therefore the total computational complexity for $P(\underline{\mu}, \mathbf{y}_1^N)$ is

$$\begin{aligned} & C\left(P(\underline{\mu}, \mathbf{y}_1^N) \text{ per possible symbol}\right) \\ &= 4 + \left[2^v\left(2^{p-v}-1\right)x + 2^{2v} + 2^{2v} + 2^v\left(2^v-1\right)x\right] \Big/ 2^p \\ &= 4 + 2^{2v+1-p} + \left(1 + 2^{2v-p} - 2^{v+1-p}\right)x, \end{aligned}$$

where $C(y)$ denotes the complexity of computing y. To compare the complexities of the original symbol-based algorithm and the first version of the simplified algorithm, we investigate their ratio,

$$\begin{aligned} & \frac{C\left(\text{Original algorithm}\right)}{C\left(\text{Simplified algorithm for } p>v\right)} \\ &= \frac{C\left(\text{Original algorithm}: \text{per possible symbol}\right)}{C\left(\text{Simplified algorithm for } p>v: \text{per possible symbol}\right)} \\ &= \frac{2^{v+1} + \left(2^v - 1\right)x}{4 + 2^{2v+1-p} + \left(1 + 2^{2v-p} - 2^{v+1-p}\right)x}. \end{aligned}$$

For $x = 6$, we compute the complexity ratios for different values of v and p in the upper right section of Table 3.1. When $v = 1$, the simplified algorithm requires more operations. Large ratios are only observed when $v >> 1$ and $p >> v$. Given $v > 1$, $p > v$ and $x \geq 0$, it is easy to prove that $2^{v+1} \geq (4+2^{2v+1-p})$ and $(2^v - 1) \geq (1 + 2^{2v-p} - 2^{v+1-p})$, and hence the complexity ratio is greater than one. In other words, this simplification will always reduce the complexity as long as $v > 1$, no matter how one measures the complexity of a table lookup operation in terms of additions. However, we see that the simplified algorithm does not reduce the complexity significantly, because when we use (3.11) to

discard the summation of $P(\underline{\mu}, s_{k-p}, s_k, \mathbf{y}_1^N)$ over s_{k-p}, we need to take more steps to compute the marginal and conditional probabilities $P(\underline{\mu}')$ and $P(\underline{\mu}''|\underline{\mu}')$ from $P(\underline{\mu})$ and hence calculate $\alpha_{k-p+v}(s_{k-p+v})$.

Table 3.1 Complexity ratios for some values of *v* and *p* with *x* = 6

v \ *p*	2	3	4	5	6	7	8	9	10
1	0.83	0.91	0.95	0.98	0.99	0.993*	0.997*	0.998*	0.999*
2	26	1.30	1.73	2.08	2.31	2.45	2.52	2.56	2.58
3	7.25	58	1.61	2.52	3.52	4.38	4.99	5.36	5.57
4	5.55	15.25	122	1.79	3.13	4.98	7.07	8.95	10.33
5	5	11.36	31.25	250	1.89	3.52	6.17	9.90	14.18
6	4.77	10.12	23	63.25	506	1.95	3.75	6.98	12.27
7	4.67	9.60	20.36	46.27	127.25	1018	1.97	3.87	7.46
8	4.62	9.37	19.26	40.84	92.82	255.25	2042	1.99	3.93

* Using higher precision to show more details

3.4.3 Complexity of the simplified algorithm for $p \le v$

After the computation of the forward and backward probabilities is completed, only two terms need to be computed in order:

1) $P(s_k, \mathbf{y}_1^N)$ using (3.17), and

2) $P(\underline{\mu}, \mathbf{y}_1^N)$ using (3.16).

The complexity of computing all 2^v states of $P(s_k, \mathbf{y}_1^N)$ is 2^v additions. For each possible symbol of $\underline{\mu}$, another (2^v -1) table lookup operations are needed to calculate $P(\underline{\mu}, \mathbf{y}_1^N)$.

$$
\begin{aligned}
&C\left(P(\underline{\boldsymbol{\mu}}, \mathbf{y}_1^N) \text{ per possible symbol}\right) \\
&\quad = C\left(\text{All states of } P(s_k, \mathbf{y}_1^N)\right) \Big/ 2^p + (2^{v-p} - 1)x \\
&\quad = 2^{v-p} + (2^{v-p} - 1)x.
\end{aligned}
$$

Compared to the original symbol-based algorithm, the complexity ratio is

$$
\frac{C\left(\text{Original algorithm}\right)}{C\left(\text{Simplified algorithm for } p \le v\right)} = \frac{2^{v+1} + \left(2^v - 1\right)x}{2^{v-p} + (2^{v-p} - 1)x}.
$$

Similarly, because $2^{v+1} \geq 2^{v-p}$ and $(2^v - 1) \geq (2^{v-p} - 1)$, this ratio is definitely greater than one, regardless of the complexity measurement of the table lookup operation. Some ratios calculated for $x = 6$ are posted in the lower left section of Table 3.1, which shows that this simplified version of the algorithm does reduce the complexity substantially.

So far, we have already investigated and compared the complexity of different versions of the symbol-based algorithm. However, one might argue that our complexity comparison in this section excludes the computation of the forward and backward probabilities, and hence might not be meaningful. In fact, we are focusing on the symbol-based algorithm for the case of multiple turbo iterations. After the initial run, the channel transition probability $P(\mathbf{y}_{k-p+1}^k \mid s_{k-p}, \underline{\boldsymbol{\mu}}, s_k)$ will not be re-computed, which means the computation of the forward and backward probabilities will become a small component of the overall computation after the second run. This fact makes the approximate analysis in this section meaningful.

3.5 Simulations on PMRCs

We employ the nonbinary LDPC coded read channel described in Section 3.1, but the turbo equalization is implemented in the recording system, as shown in Fig. 3.4.

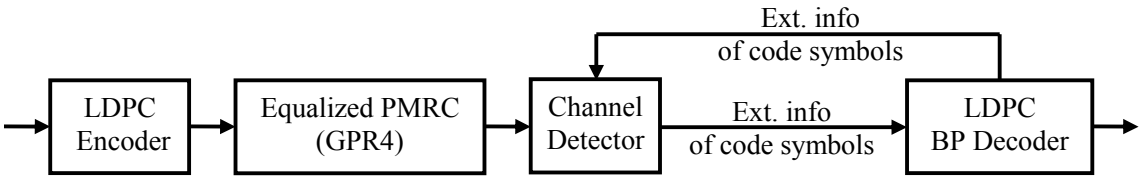


Fig. 3.4. A nonbinary LDPC coded PMR system with turbo equalization.

The nonbinary LDPC coded system is simulated with three different channel detectors: the original bit-based BCJR algorithm, Hoeher's OBBD, and the symbol-based algorithm derived in this chapter. The nonbinary LDPC BP decoder is set to perform at most 50 iterations. For comparison, the system is simulated both with and without turbo equalization, which is implemented with at most three turbo iterations. As shown in Fig. 3.5, the symbol-based algorithm provides a considerably large coding gain compared to the bit-based BCJR algorithm. The performance of Hoeher's OBBD and the symbol-based algorithm running only once are identical, which verifies that, without turbo equalization, Hoeher's OBBD is theoretically equivalent to our optimal symbol-based algorithm. With at most three turbo iterations, Hoeher's OBBD does not improve the performance, while the symbol-based algorithm provides an additional coding gain. To further highlight the performance gap between the OBBD and the optimal symbol-based algorithm, we can reduce the maximum number of BP iterations and allow more channel iterations. Simulation results with at most ten BP iterations and six turbo iterations are shown in Fig. 3.6, where the symbol-based algorithm achieves 0.2 dB gain over the OBBD.

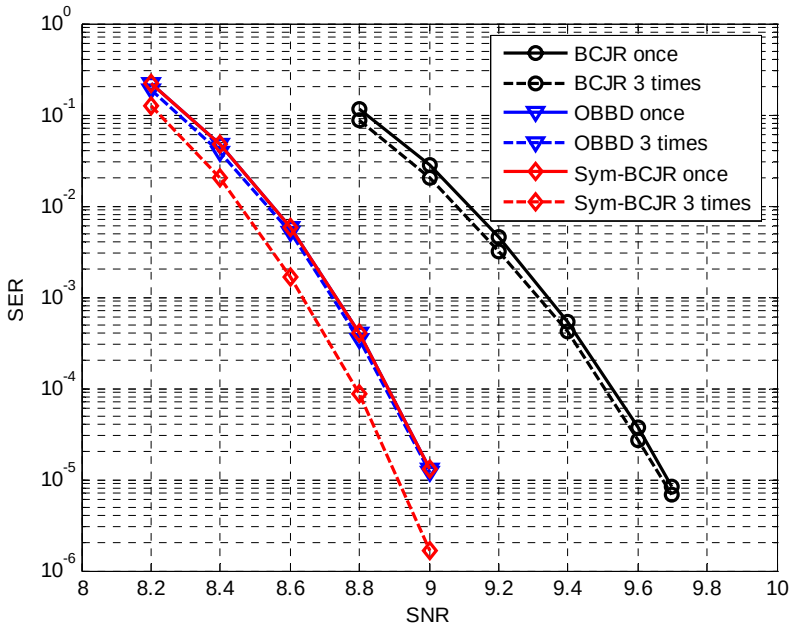


Fig. 3.5. Performance of nonbinary LDPC coded PMRCs with different channel detectors and a maximum of 50 BP iterations.

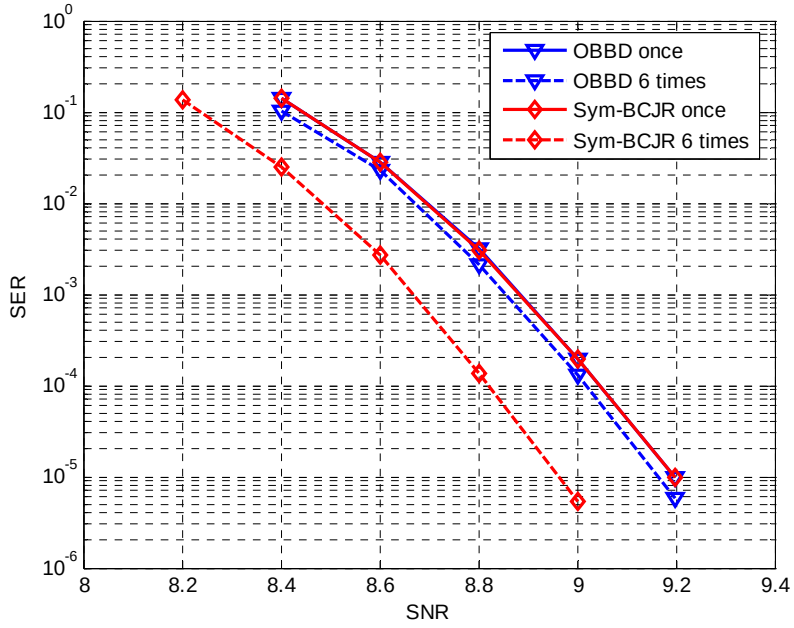


Fig. 3.6. Performance of nonbinary LDPC coded PMRCs with different channel detectors and a maximum of 10 BP iterations.

4 Improved BP Decoders for LDPC Coded PR Channels

The parity-check matrix of an LDPC code can be expressed as a factor graph [43], and a message passing algorithm, such as the BP algorithm, which finds the marginal probabilities of variable nodes on factor graphs, can be used as an efficient decoder for LDPC codes. It is well known that BP decoding is not optimal (or exact) unless two particular conditions are satisfied. One condition is that the factor graph must be tree-like, i.e., cycle-free. Unfortunately, finite length LDPC codes always have cycles on their factor graphs, making the messages passed on the graphs dependent on each other, after a few BP iterations. A common approach is to eliminate all short cycles plus those cycles that may have some harmful properties during code construction.

The other necessary condition for optimal BP decoding is that the intrinsic information [51] of variable nodes should be independent of each other. The intrinsic information is called the initial information in [7], and is simply the soft information at the input of the BP decoder, which is continuously used in all BP iterations. Sometimes this condition is overlooked, and in some situations it may not be an issue. One such example is the LDPC coded AWGN channel, $y_i = x_i + n_i$, with $n_i \sim N(0,\sigma^2)$. It is well known that the channel messages, in terms of LLRs, $L_i = 2y_i / \sigma^2$, are independent of each other. However, this is not the case for PR channels; the LLRs output by the channel detector are usually correlated, and this has been a source of concern for many years. Evidence to that effect is the OBBD we have discussed in Chapter 3, which generates the probability distributions for subblocks in the channel sequence. Therefore, for an LDPC coded PR channel, the BP decoder is passing correlated messages and becomes suboptimal from the very first

iteration, leading to a degradation of its decoding performance.

One would expect to get some coding gains if we exploit the correlations between channel messages during BP decoding. Successful examples of such an approach are the use of the OBBD and the new symbol-based BCJR on nonbinary LDPC coded PR channels, which we introduced in Chapter 3. Each code symbol in a nonbinary LDPC codeword corresponds to a subblock of the binary channel sequence. The channel detector provides the probability distributions of code symbols for the nonbinary BP decoding, where bits in the same code symbol are always considered statistically correlated. However, bits in different code symbols are still assumed to be independent, which is not actually true.

In this chapter, we introduce an improved BP (IBP) decoder to take into account the correlations between channel messages inspired by the coded modulation BP (CMBP) algorithm proposed in [52]. The CMBP algorithm is designed for an LDPC coded discrete memoryless channel (DMC) with a multilevel modulated signal (MMS). The nature of this channel makes the bits in the LDPC codewords independent, unless they are related to the same modulated channel symbol. One may argue that it would be better to code this channel with a nonbinary LDPC code, whose code symbol has the same size as the modulated channel symbol, but that is not the focus of this work. From our perspective, the most interesting part of the CMBP algorithm is that it modifies the binary BP algorithm to utilize the dependence between channel messages. We find that a similar idea can be used on PR channels where the channel messages are also correlated but not separated by code or modulation symbols.

This chapter is organized as follows. In Section 4.1, we give a complete derivation of the

IBP decoder for binary LDPC coded PR channels. In Section 4.2, we apply the IBP decoder to an ideal PR channel, and investigate the relationship between the performance of the IBP decoder and the correlations among channel messages. In Section 4.3, we consider the application of the IBP decoder to PMRCs. In Section 4.4, we further extend the IBP algorithm to the decoding of nonbinary LDPC codes. In Section 4.5, we evaluate the performance of the improved nonbinary BP decoding on PMRCs. In Section 4.6, we investigate the implementation of turbo equalization for the IBP decoder, and we conclude this chapter with a discussion of the results in Section 4.7.

4.1 Improved BP decoding

Consider a block of N bits, $\boldsymbol{\mu}_1^N \triangleq (\mu_1, \mu_2, \cdots \mu_N)$, transmitted through a PR channel and the observed noisy channel output, $\boldsymbol{y}_1^N \triangleq (y_1, y_2, \cdots y_N)$. In order to get the minimum bit error rate, we use the BCJR algorithm as the channel detector. At the detector output, we obtain a sequence of soft channel messages (LLRs), $\boldsymbol{L}_1^N \triangleq (L_1, L_2, \cdots L_N)$, corresponding to the bits in the input block. We assume that any pair of channel messages has a strong dependence only if they are within a relatively small distance; in other words, the dependence between channel messages vanishes, as they get far apart. This assumption is intuitively reasonable and will be validated in Section 4.2.

Given that the channel input sequence is LDPC coded, when the LDPC decoder is handling a channel message L_i, it is expected to consider the dependence of L_i on the c channel messages before L_i and the ac channel messages after L_i, as shown in Fig. 4.1, where c stands for the causal length and ac for the anti-causal length. In other words, when the LDPC decoder is processing a channel message (except the ones near the block

boundary), there will be $p = c+ac+1$ channel messages considered at the same time. The dependence among the p consecutive channel messages will be used in LDPC decoding, in terms of the joint probability distribution of the corresponding p bits in the transmitted sequence.

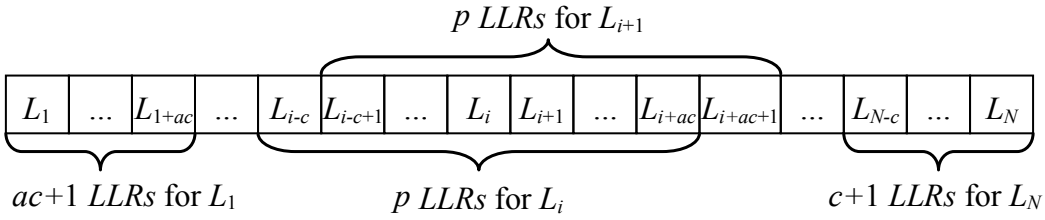


Fig. 4.1. Dependence between channel messages (LLRs) is expressed as the joint distribution of p consecutive bits, i.e., the distribution of p-bit subblocks. These subblocks overlap and become shorter near the channel block boundaries.

4.1.1 Improved BP algorithm for PR channels

The IBP algorithm is essentially based on the same idea as the CMBP algorithm in [52], except that we always consider a channel message correlated with the ones before and after it, not just in the same modulation symbol. We also provide a detailed analysis and derivation, which clarifies the assumptions and approximations, used in the algorithm and make the later extension to the nonbinary BP decoder straightforward.

Let $\underline{\boldsymbol{\mu}} \triangleq \boldsymbol{\mu}_{k-p+1}^{k}$ be the channel input subblock from time k–p+1 to time k, where the corresponding channel messages at the output of the channel detector are assumed to be dependent on each other. For convenience, we further use $\underline{\boldsymbol{\mu}}^i \triangleq \mu_{k-p+i}$ with $i \in \{1,\cdots,p\}$ to denote the bits in $\underline{\boldsymbol{\mu}}$, and let $\underline{\boldsymbol{\mu}}^{(i)}$ stand for all bits in $\underline{\boldsymbol{\mu}}$ except $\underline{\boldsymbol{\mu}}^i$. We are interested in the *a posteriori* probability (APP) for the particular bit $\underline{\boldsymbol{\mu}}^{c+1}$: $P(\underline{\boldsymbol{\mu}}^{c+1} | \boldsymbol{y}_1^N, \mathbf{H})$, where $\mathbf{H}$ is

the parity-check matrix of the LDPC code. However, it is not easy to compute such APP for any bit conditioned on the whole parity-check matrix **H**; actually, BP is a way to get an approximation on such APP with the assumption that channel messages are independent. To make things easier but keep the channel message dependence within $\underline{\boldsymbol{\mu}}$, we compute the APP $P(\underline{\boldsymbol{\mu}}^{c+1} \mid \boldsymbol{y}_1^N,\ \underline{\boldsymbol{z}} = \underline{0})$ at each BP iteration, where $\underline{\boldsymbol{z}}$ represents all parity-checks corresponding to the subblock $\underline{\boldsymbol{\mu}}$, and $\underline{\boldsymbol{z}}^i$ denotes the parity-checks related to the bit $\underline{\boldsymbol{\mu}}^i$. Then this APP can be expanded as follows,

$$
\begin{aligned}
&P\left(\underline{\boldsymbol{\mu}}^{c+1} \mid \boldsymbol{y}_1^N, \underline{\boldsymbol{z}} = \underline{0}\right) \\
&\quad = \sum_{\underline{\boldsymbol{\mu}}^{(c+1)}} P\left(\underline{\boldsymbol{\mu}} \mid \boldsymbol{y}_1^N, \underline{\boldsymbol{z}} = \underline{0}\right) \\
&\quad = \sum_{\underline{\boldsymbol{\mu}}^{(c+1)}} \left[\frac{p\left(\underline{\boldsymbol{\mu}}\right) P\left(\underline{\boldsymbol{z}} = \underline{0} \mid \underline{\boldsymbol{\mu}}\right) P\left(\boldsymbol{y}_1^N \mid \underline{\boldsymbol{z}} = \underline{0}, \underline{\boldsymbol{\mu}}\right)}{p\left(\boldsymbol{y}_1^N, \underline{\boldsymbol{z}} = \underline{0}\right)} \right] \\
&\quad = \sum_{\underline{\boldsymbol{\mu}}^{(c+1)}} \left[\frac{p\left(\underline{\boldsymbol{\mu}}\right) P\left(\underline{\boldsymbol{z}} = \underline{0} \mid \underline{\boldsymbol{\mu}}\right) P\left(\boldsymbol{y}_1^N \mid \underline{\boldsymbol{\mu}}\right)}{p\left(\boldsymbol{y}_1^N\right) p\left(\underline{\boldsymbol{z}} = \underline{0}\right)} \right] \\
&\quad = \sum_{\underline{\boldsymbol{\mu}}^{(c+1)}} \left[\frac{P\left(\underline{\boldsymbol{\mu}} \mid \boldsymbol{y}_1^N\right) \cdot P\left(\underline{\boldsymbol{\mu}} \mid \underline{\boldsymbol{z}} = \underline{0}\right)}{p\left(\underline{\boldsymbol{\mu}}\right)} \right] \\
&\quad = \frac{1}{p\left(\underline{\boldsymbol{\mu}}\right)} \sum_{\underline{\boldsymbol{\mu}}^{(c+1)}} \left[P\left(\underline{\boldsymbol{\mu}} \mid \boldsymbol{y}_1^N\right) \cdot P\left(\underline{\boldsymbol{\mu}} \mid \underline{\boldsymbol{z}} = \underline{0}\right) \right] \\
&\quad = \frac{1}{p\left(\underline{\boldsymbol{\mu}}\right)} \sum_{\underline{\boldsymbol{\mu}}^{(c+1)}} \left[P\left(\underline{\boldsymbol{\mu}} \mid \boldsymbol{y}_1^N\right) \cdot \prod_{i=1}^{p} P\left(\underline{\boldsymbol{\mu}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0}\right) \right],
\end{aligned}
\tag{4.1}
$$

where $P\left(\underline{\boldsymbol{\mu}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0}\right)$ is the APP for bit $\underline{\boldsymbol{\mu}}^i$ computed in one BP iteration. Note that $P\left(\underline{\boldsymbol{\mu}} \mid \underline{\boldsymbol{z}} = \underline{0}\right) = \prod_{i=1}^{p} P\left(\underline{\boldsymbol{\mu}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0}\right)$ is only true for the LDPC codes with tree-like graphs; but it is a good approximation given that there is no cycle-4 on the sub-graph corresponding to the subblock $\underline{\boldsymbol{\mu}}$. During the derivation of (4.1), we made use of the

independence between $\boldsymbol{y}_1^N$ and $\underline{\boldsymbol{z}}$. In addition, we assume $p(\underline{\boldsymbol{\mu}})$ to be uniformly distributed. Then the *a posteriori* LLR message for bit $\underline{\boldsymbol{\mu}}^{c+1}$ can be expressed as

$$\begin{aligned}
V\left(\underline{\boldsymbol{\mu}}^{c+1}\right) &\triangleq \log \frac{P\left(\underline{\boldsymbol{\mu}}^{c+1}=0 \mid \boldsymbol{y}_1^N, \underline{\boldsymbol{z}}=\underline{0}\right)}{P\left(\underline{\boldsymbol{\mu}}^{c+1}=1 \mid \boldsymbol{y}_1^N, \underline{\boldsymbol{z}}=\underline{0}\right)} \\
&= \log \frac{\sum_{\underline{\boldsymbol{\mu}}^{(c+1)}}\left[P\left(\underline{\boldsymbol{\mu}}^{(c+1)}, \underline{\boldsymbol{\mu}}^{c+1}=0 \mid \boldsymbol{y}_1^N\right)\prod_{j=1, j\neq c+1}^{p} P\left(\underline{\boldsymbol{\mu}}^{j} \mid \underline{\boldsymbol{z}}^{j}=\underline{0}\right)\right]}{\sum_{\underline{\boldsymbol{\mu}}^{(c+1)}}\left[P\left(\underline{\boldsymbol{\mu}}^{(c+1)}, \underline{\boldsymbol{\mu}}^{c+1}=1 \mid \boldsymbol{y}_1^N\right)\prod_{j=1, j\neq c+1}^{p} P\left(\underline{\boldsymbol{\mu}}^{j} \mid \underline{\boldsymbol{z}}^{j}=\underline{0}\right)\right]} \\
&\quad + \log \frac{P\left(\underline{\boldsymbol{\mu}}^{c+1}=0 \mid \underline{\boldsymbol{z}}^{c+1}=\underline{0}\right)}{P\left(\underline{\boldsymbol{\mu}}^{c+1}=1 \mid \underline{\boldsymbol{z}}^{c+1}=\underline{0}\right)} \\
&= \log \frac{\sum_{\underline{\boldsymbol{\mu}}^{(c+1)}}\left[P\left(\underline{\boldsymbol{\mu}}^{(c+1)} \mid \underline{\boldsymbol{\mu}}^{c+1}=0, \boldsymbol{y}_1^N\right)\prod_{j=1, j\neq c+1}^{p} P\left(\underline{\boldsymbol{\mu}}^{j} \mid \underline{\boldsymbol{z}}^{j}=\underline{0}\right)\right]}{\sum_{\underline{\boldsymbol{\mu}}^{(c+1)}}\left[P\left(\underline{\boldsymbol{\mu}}^{(c+1)} \mid \underline{\boldsymbol{\mu}}^{c+1}=1, \boldsymbol{y}_1^N\right)\prod_{j=1, j\neq c+1}^{p} P\left(\underline{\boldsymbol{\mu}}^{j} \mid \underline{\boldsymbol{z}}^{j}=\underline{0}\right)\right]} \\
&\quad + \log \frac{P\left(\underline{\boldsymbol{\mu}}^{c+1}=0 \mid \boldsymbol{y}_1^N\right)}{P\left(\underline{\boldsymbol{\mu}}^{c+1}=1 \mid \boldsymbol{y}_1^N\right)} + \log \frac{P\left(\underline{\boldsymbol{\mu}}^{c+1}=0 \mid \underline{\boldsymbol{z}}^{c+1}=\underline{0}\right)}{P\left(\underline{\boldsymbol{\mu}}^{c+1}=1 \mid \underline{\boldsymbol{z}}^{c+1}=\underline{0}\right)} \\
&= \log \frac{\sum_{\underline{\boldsymbol{\mu}}^{(c+1)}}\left[P\left(\underline{\boldsymbol{\mu}}^{(c+1)} \mid \underline{\boldsymbol{\mu}}^{c+1}=0, \boldsymbol{y}_1^N\right)\prod_{j=1, j\neq c+1}^{p} P\left(\underline{\boldsymbol{\mu}}^{j} \mid \underline{\boldsymbol{z}}^{j}=\underline{0}\right)\right]}{\sum_{\underline{\boldsymbol{\mu}}^{(c+1)}}\left[P\left(\underline{\boldsymbol{\mu}}^{(c+1)} \mid \underline{\boldsymbol{\mu}}^{c+1}=1, \boldsymbol{y}_1^N\right)\prod_{j=1, j\neq c+1}^{p} P\left(\underline{\boldsymbol{\mu}}^{j} \mid \underline{\boldsymbol{z}}^{j}=\underline{0}\right)\right]} \\
&\quad + L\left(\underline{\boldsymbol{\mu}}^{c+1}\right) + U\left(\underline{\boldsymbol{\mu}}^{c+1}\right),
\end{aligned} \tag{4.2}$$

where $L(\underline{\boldsymbol{\mu}}^{c+1}) \triangleq \log\ [P(\underline{\boldsymbol{\mu}}^{c+1}=0 \mid \boldsymbol{y}_1^N)/P(\underline{\boldsymbol{\mu}}^{c+1}=1 \mid \boldsymbol{y}_1^N)]=L_{k-p+c+1}$ is the channel message computed by the channel detector. $U(\underline{\boldsymbol{\mu}}^{c+1}) \triangleq \log\ [P(\underline{\boldsymbol{\mu}}^{c+1}=0 \mid \underline{\boldsymbol{z}}^{c+1}=\underline{0})/P(\underline{\boldsymbol{\mu}}^{c+1}=1 \mid \underline{\boldsymbol{z}}^{c+1}=\underline{0})]$ $= \sum_{i\in N(\underline{\boldsymbol{\mu}}^{c+1})} U(i \rightarrow \underline{\boldsymbol{\mu}}^{c+1})$, where

$$U(i \rightarrow \underline{\boldsymbol{\mu}}^{c+1}) \triangleq 2 \cdot \tanh^{-1}\left(\prod_{l\in N(i)\setminus \underline{\boldsymbol{\mu}}^{c+1}} \tanh\left(V(l \rightarrow i)/2\right)\right) \tag{4.3}$$

is the information passed from check i to bit $\underline{\boldsymbol{\mu}}^{c+1}$, and $V(l \rightarrow i)$ is the information sent from

bit l to check i. For $l=\underline{\mu}^{c+1}$, we have

$$V(\underline{\mu}^{c+1}\to i)\triangleq V(\underline{\mu}^{c+1})-U(i\to\underline{\mu}^{c+1})\,. \tag{4.4}$$

Note that, for convenience, we are abusing the bit index notation, i.e., "bit $\underline{\mu}^{c+1}$" represents the same bit as "bit $k-p+c+1$". As in the CMBP algorithm, the check-to-bit information here is computed in the same way as in the BP algorithm, while the dependence between channel messages is considered in the bit-to-check information. Clearly, if all channel messages are independent of each other, the first term in (4.2) is equal to zero and the iterative algorithm defaults to the original BP algorithm.

The factorization in (4.2) is useful to understand the main idea of the IBP algorithm, but it is not computationally efficient. We note that $\prod_{j=1,j\neq c+1}^{p}P\left(\underline{\mu}^{j}\mid\underline{z}^{j}=\underline{0}\right)=\exp\left(\sum_{j=1,j\neq c+1}^{p}\left(1-\underline{\mu}^{j}\right)U(\underline{\mu}^{j})\right)\Big/\alpha$, where $\alpha=\prod_{j=1,j\neq c+1}^{p}\left\{1+\exp\left[U(\underline{u}^{j})\right]\right\}$ is independent of the sum over $\underline{\mu}^{(c+1)}$. Given that the OBBD provides the distributions $P\left(\underline{\mu}\mid\underline{y}_1^N\right)$, the *a posteriori* LLR message of (4.2) can be more efficiently computed as

$$V\left(\underline{\mu}^{c+1}\right)=\log\frac{\sum_{\underline{\mu}^{(c+1)}}\left[P\left(\underline{\mu}^{(c+1)},\underline{\mu}^{c+1}=0\mid\underline{y}_1^N\right)\exp\left(\sum_{j=1,j\neq c+1}^{p}\left(1-\underline{\mu}^{j}\right)U(\underline{\mu}^{j})\right)\right]}{\sum_{\underline{\mu}^{(c+1)}}\left[P\left(\underline{\mu}^{(c+1)},\underline{\mu}^{c+1}=1\mid\underline{y}_1^N\right)\exp\left(\sum_{j=1,j\neq c+1}^{p}\left(1-\underline{\mu}^{j}\right)U(\underline{\mu}^{j})\right)\right]}+U(\underline{\mu}^{c+1}). \tag{4.5}$$

4.1.2 Channel detector for IBP

In order to execute the IBP decoding algorithm, the channel detector must compute the distributions $P\left(\underline{\mu}\mid\underline{y}_1^N\right)$. Actually, the OBBD proposed in [48] does just that. Note that in

Chapter 3, the OBBD was used on nonbinary LDPC coded PR channels to generate probability distributions for separate subblocks; but for IBP, we want the probability distributions for overlapped subblocks. For example, for an N-bit channel block, probability distributions for at least $N–p+1$ overlapped subblocks will be computed, as shown in Fig. 4.1. (We may need additional probability distributions at block boundaries.) No matter what kind of distribution we want, for separate subblocks or for overlapped subblocks, its computation for a particular subblock is the same. One thing which needs to be pointed out is that the OBBD also needs to generate the LLR for each bit as in the original BCJR algorithm, because the IBP algorithm uses the channel message L_l to initialize the bit-to-check information $V(l \rightarrow i)$.

4.1.3 Boundary management

In the derivation of the IBP algorithm, we assume a channel message to be correlated with $c + ac = p - 1$ channel messages around it. Since the LDPC coded channel block is of finite length, the channel messages near the block boundaries may be dependent on fewer than p-1 other messages, as shown in Fig. 4.1. At the left boundary, L_i with $i<c$ only depends on $\{L_1, \ldots, L_{i-1}\}$ and $\{L_{i+1}, \ldots, L_{i+ac}\}$. Similarly, at the right boundary, L_i with $i>N–ac$ only depends on $\{L_{i-c}, \ldots, L_{i-1}\}$ and $\{L_{i+1}, \ldots, L_N\}$. We manage the OBBD to generate probabilities for superposed subblocks with lengths from $ac + 1$ to p at the left boundary and with lengths from p to $c+1$ at the right boundary. Meanwhile, the IBP iteration is modified to take fewer channel messages into account at the block boundaries, during the computation of (4.2). The modification is straightforward.

4.2 LDPC coded PR channel

We design an LDPC code using the PEG algorithm [50]; the null space of a 456 by 4551 parity-check matrix, with a constant column weight of four, gives a (4551, 4096) LDPC code. The code rate is around 0.9. The reason for choosing an LDPC code with such a high rate is that we are considering applying the IBP decoding to equalized PMRCs in the next section; for consistency, we will use the same LDPC code on both PR channels and PMRCs. We use a very simple detection and decoding architecture: for each sector, the channel detector runs only once, then the BP or IBP decoders operate on the information sent from the detector; no information is fed back to the channel detector.

As mentioned at the beginning of this chapter, the channel messages for the PR channel are not independent. To investigate the dependence among them, we simulate the channel and estimate the autocorrelation sequence of the LLRs at the BCJR detector output, where the implicit assumption is that the blocks of channel messages are wide-sense stationary random sequences. However, we are not interested in proving the validity of the stationarity assumption. Our experiments will show whether the autocorrelation sequence is a valid indicator of the dependence between channel messages.

We consider as an example the EPR4 channel with integer coefficients $1+D-D^2-D^3$ on AWGN. This PR channel used to be one of the common equalization targets for longitudinal magnetic recording channels. The SNR is defined by $\mathrm{SNR}=\sum_k f_k^2/\sigma^2$, where $\{f_k\}$ are the coefficients of the channel response and σ^2 is the variance of the AWGN. The channel input bits are modulated as NRZ signals before transmission. For the LDPC coded EPR4 channel, we draw the autocorrelation sequences of the channel messages generated by the BCJR detector in Fig. 4.2.

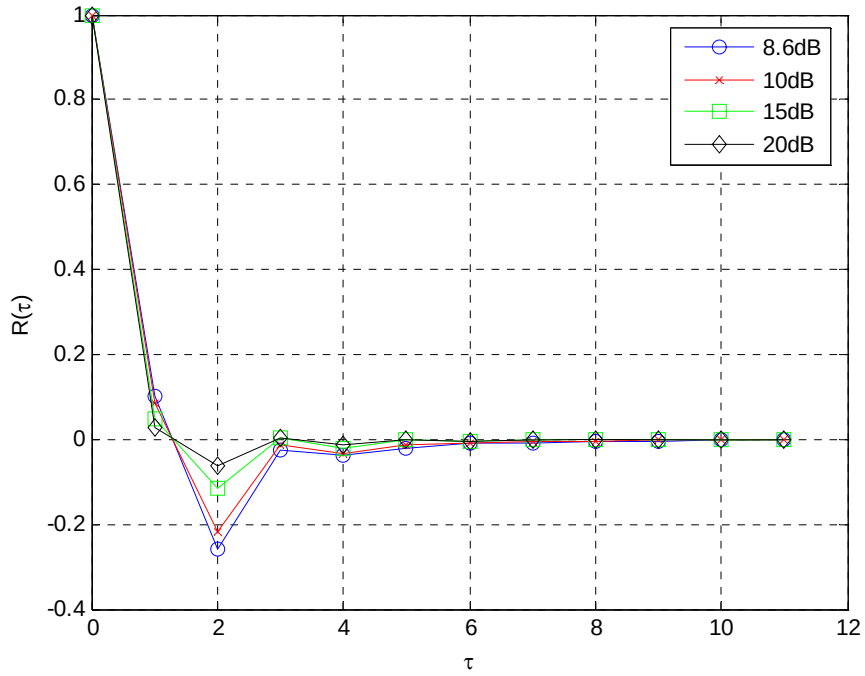


Fig. 4.2. Normalized autocorrelation sequences of channel messages for the LDPC coded EPR4 channel.

We can see that the correlation between two channel messages is only significant within a small range; it vanishes with increasing time lag. This fact validates the assumption used in the derivation of the IBP algorithm. In addition, it is intuitive that the correlations are smaller for channel messages observed at higher SNRs; on a noise-free channel, the BCJR detector will give a white sequence of LLRs which are either negative infinity or infinity.

We apply the IBP decoder with at most fifty iterations to an LDPC coded EPR4 channel with the expectation of getting better performance from the utilization of the correlations among channel messages. However, it is important to choose appropriate values for c and ac in the IBP algorithm. Since $p=c+ac+1$ determines the complexity of both the IBP decoder and the OBBD channel detector, we seek good performance with a relatively small p. Therefore, we design an experiment to test the performance of the IBP decoding with

different choices of p and c. As shown in Fig. 4.3, the channel is simulated at an SNR of 8.6 dB, with p varying from 1 to 7 and c from 0 to p-1. Note that, for p=1, IBP decoding is the original BP algorithm. As expected, we always have a chance to get better performance with larger p. Note that for a given p the performance curve is symmetric, i.e., IBP decoders with $c=i$ and $c=p-i-1$, with $0 \le i < p$ give almost the same performance, which means that a channel message has the same correlation with messages at the same distance.

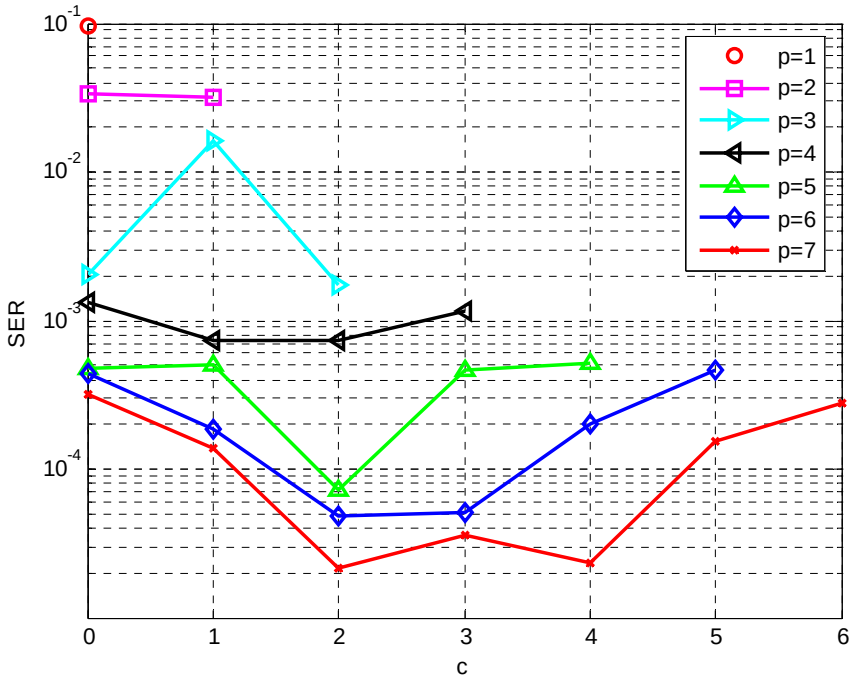


Fig. 4.3. Performance test of LDPC coded and IBP decoded EPR4 channel.

The non-monotonic performance behavior for different values of c with a given p is quite interesting, and can be explained by the autocorrelation sequence of the channel messages. For example, for p=3, we get much worse performance with c=1 than with c=0 or c=2. From Fig. 4.2, we can see that at SNR=8.6 dB, $|R(2)|>|R(1)|$, which means that a channel message has a weaker dependence on its adjacent messages than on the ones two bit intervals away from it. Therefore, for the IBP decoders with p=3, choosing c=0 or c=2 can

help the decoding algorithm take into account the more dependent channel messages, and hence achieve better performance. Another example is that for p=7, c=3 we get slightly worse performance than for c=2 or c=4, because $|R(4)|>|R(3)|$. Performance curves for other values of p can be explained similarly. Furthermore, it is intuitive that, in order to get good performance, it is necessary to let the sliding window of a channel message include the messages that have the strongest correlations. In Fig. 4.3, with p varying from 1 to 7, the largest performance improvements occur at p=3 with c=0 or c=2 as well as p=5 with c=2, since $|R(2)|>|R(\tau)|$ for all $\tau>0$ and $\tau\neq 2$. Finally, we plot the performance curves for the best choice for each value of p, in terms of sector (block) error rate (SER) in Fig. 4.4, and show that the IBP decoding provides gains as large as 0.7 dB over the original BP decoding. (We measure the performance gains at the SER of 10^{-5}.)

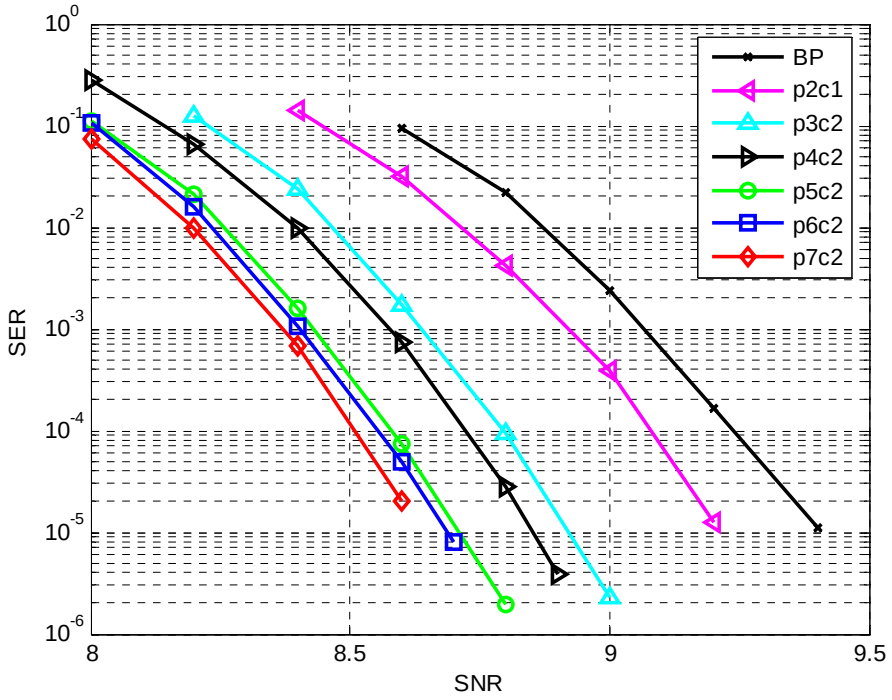


Fig. 4.4. Performance of LDPC coded and IBP decoded EPR4 channel.

Before we conclude this section, we would like to point out that we find the correlations among channel messages to be closely related to the response of the PR channel. Since equalized PMRCs have optimized generalized PR (GPR) targets with real coefficients and the magnitudes of the target responses usually decay with time, we consider another ideal GPR4 channel with a response $1+0.5D+0.2D^2+0.05D^3$, which is assumed to be coded by the same LDPC code used for the EPR4 channel. We show the normalized autocorrelation sequences of the channel messages for the GPR4 channel in Fig. 4.5. For $\tau \leq 4$, larger values of τ have weaker correlation, which means that the best performance of IBP decoding for a given p will always occur with $c = \lfloor (p-1)/2 \rfloor$. We will not present performance results for the GPR4 channel here, because similar results are given in the next section.

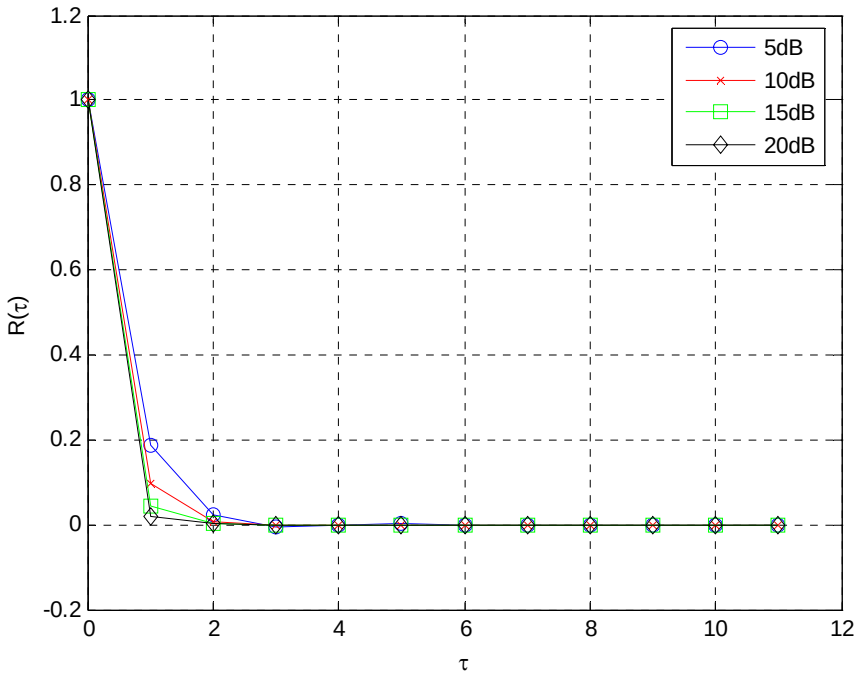


Fig. 4.5. Normalized autocorrelation sequences of channel messages for the LDPC coded GPR4 channel.

4.3 LDPC coded PMRCs

In this section, we consider the application of IBP decoding to equalized PMRCs, coded with the same PEG-designed LDPC code as in Section 4.2. We use a mix of 90% jitter noise power and 10% electronics noise power in all simulations. This channel is equalized to optimized GPR4 targets. A low user density D_u = 0.8741 and a high user density D_u = 1.2238 will be considered in this chapter.

In this section, we are still using a simple detection and decoding architecture, in which there is no feedback from the LDPC decoder to the input of the channel detector, and we perform at most fifty BP or IBP iterations. As we did in Section 4.2, the autocorrelation sequences of the channel messages for both user densities are estimated first and drawn in Fig. 4.6. At SNR=4.5 dB for D_u = 0.8741 and SNR=8.8 dB for D_u = 1.2238 the performance of the traditional BP decoding is in the waterfall region. We will also test the performance results at these SNRs, later. In the relevant range ($\tau \leq 4$) of Fig. 4.6, both autocorrelation sequences have decaying magnitude, which means $c = \lfloor (p-1)/2 \rfloor$ will always be the best choice for a given p. In addition, the channel messages on the channel with lower user density have relatively weaker correlations. Note that the PMRCs will never be perfectly equalized; at higher user density, the larger equalization error and stronger correlated and data dependent noise cause more severe intersymbol interference (ISI). It is reasonable that higher ISI lead to stronger correlations among channel messages.

We test the performance of the channel with D_u = 0.8741 at SNR=4.5 dB and show the results in Fig. 4.7. Since $|R(\tau)|$ with $\tau \geq 2$ in Fig. 4.6 is very small, for p=5 and c=2 and for larger values of p we only get marginal performance improvements and hence p=3 and c=1 is an appropriate choice for good performance and low complexity. We show the

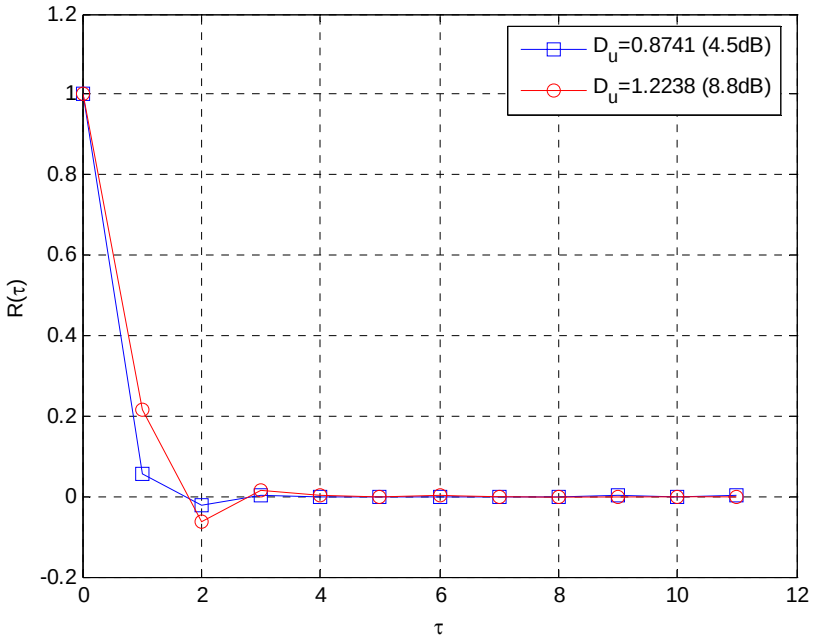


Fig. 4.6. Normalized autocorrelation sequences of channel messages for LDPC coded PMRCs.

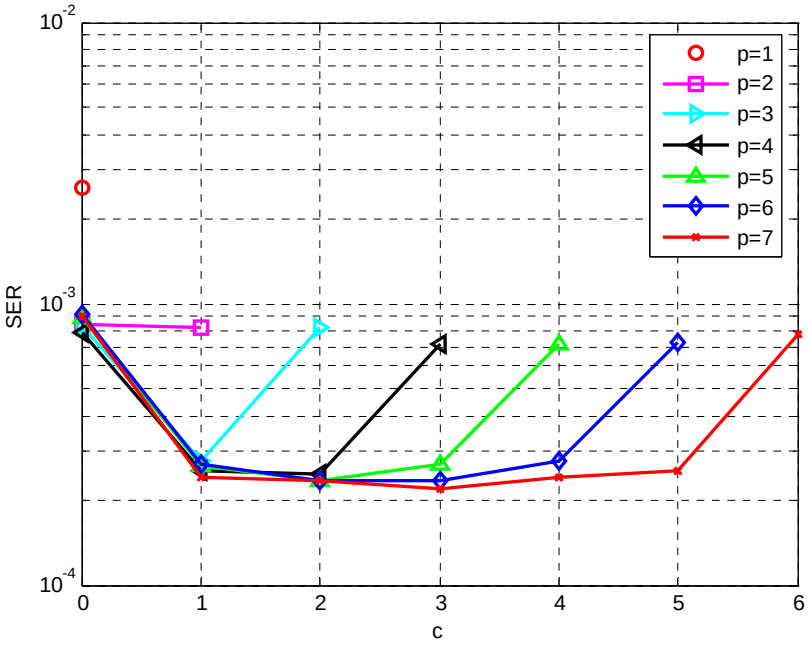


Fig. 4.7. Performance test of LDPC coded and IBP decoded PMRCs with D_u=0.8741, at SNR=4.5dB.

performance of IBP decoding with p=2 and c=1, p=3 and c=1, p=5 and c=2, and p=7 and c=3 in Fig. 4.8, where the largest gain over BP is about 0.18 dB. Compared with the simulation for the EPR4 channel, it is clear that IBP decoding cannot provide large gains with weakly correlated channel messages.

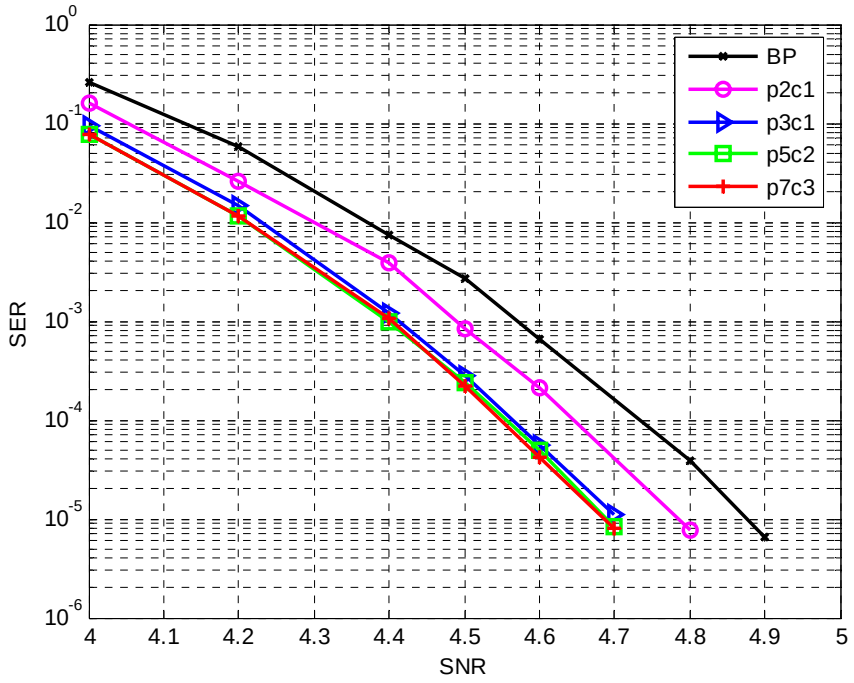


Fig. 4.8. Performance of LDPC coded and IBP decoded PMRCs with D_u = 0.8741.

For the PMRC with D_u = 1.2238, we do the performance test at SNR=8.8 dB and show the results in Fig. 4.9, where the relationship between the autocorrelation of channel messages and the performance of IBP decoding is verified once again. We show the performance of IBP decoding with p=2 and c=1, p=3 and c=1, p=5 and c=2, and p=7 and c=3 in Fig. 4.10, and IBP decoding provides gains as large as 0.6 dB over BP decoding. Our understanding is that high correlations among channel messages due to higher user density make it possible for IBP decoding to provide larger gains.

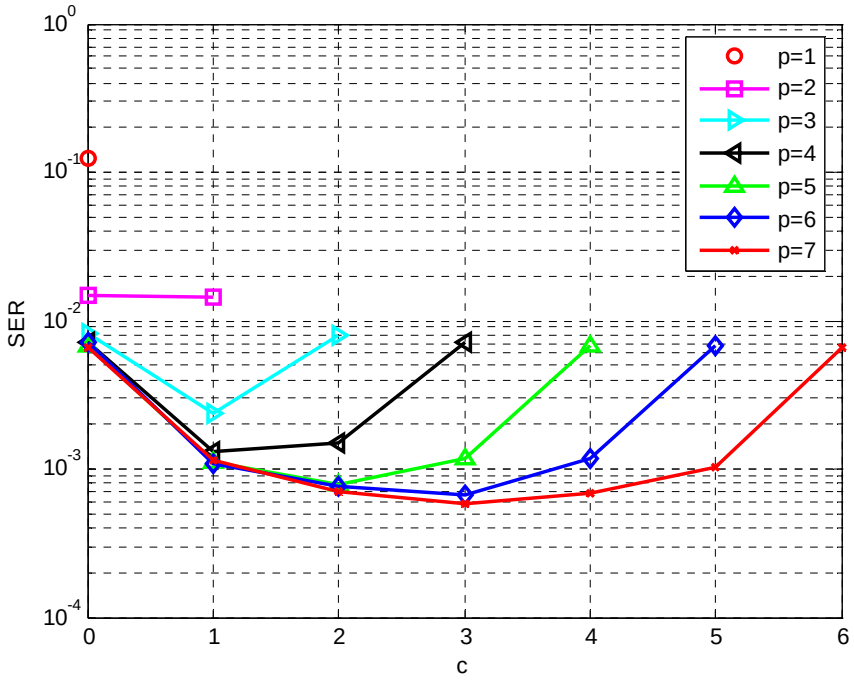


Fig. 4.9. Performance test of LDPC coded and IBP decoded PMRCs with D_u=1.2238, at SNR=8.8 dB.

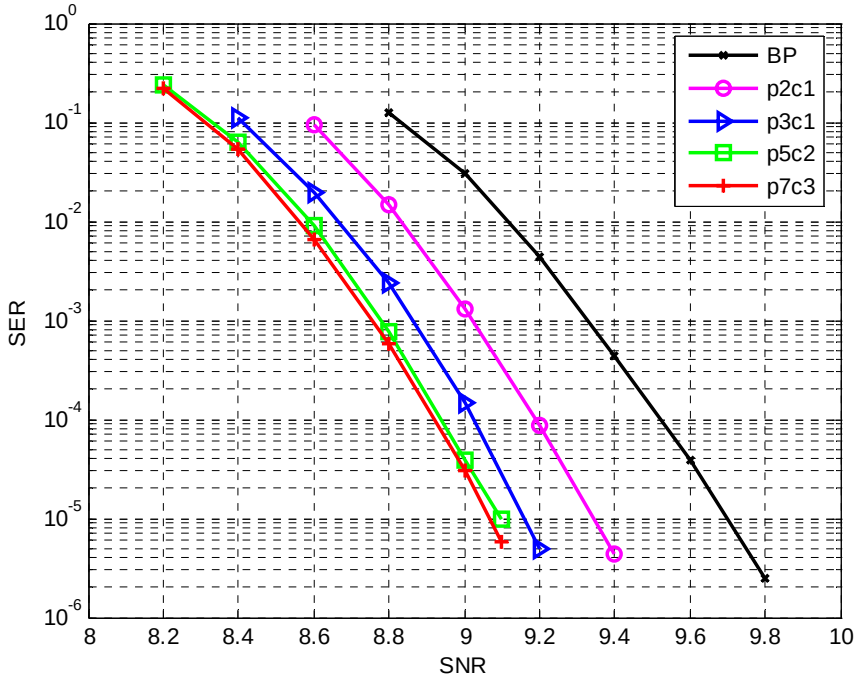


Fig. 4.10. Performance of LDPC coded and IBP decoded PMRCs with D_u = 1.2238.

4.4 Improved nonbinary BP decoding

It is well known that nonbinary LDPC coded channels can outperform binary LDPC coded channels [38], especially for those with severe ISI [53], [54]. We expect to get additional gains by extending the IBP technique to nonbinary BP decoding.

4.4.1 Improved nonbinary BP algorithm for PR channels

Consider a binary-input PR channel, where the channel inputs are nonbinary LDPC codewords. Given that the nonbinary LDPC code is over GF(q) with $q=2^p$, each code symbol consists of p consecutive bits; and each N-bit-long codeword contains $N_s=N/p$ code symbols, which are denoted by $\boldsymbol{x}_1^{N_S} \triangleq (x_1, \ldots, x_{N_S})$, where $x_n = \boldsymbol{\mu}_{(n-1)p+1}^{np}$. The channel message for the code symbol x_n is represented by $q_n^{(Ch)}$, which is a probability distribution of x_n. As in the binary case, we assume that the channel message for a code symbol is correlated with the c' messages before it and the ac' messages after it, and hence there are $p'=c'+ac'+1$ channel messages considered at the same time. Let $\underline{\boldsymbol{x}} \triangleq \boldsymbol{x}_{n-c'}^{n+ac'}$ denote the p' code symbols in the sliding window for x_n, $\underline{\boldsymbol{x}}^i \triangleq x_{n-c'+i}$ with $i \in \{0, \ldots, p'-1\}$ be the i-th symbol in $\underline{\boldsymbol{x}}$, and $\underline{\boldsymbol{x}}^{(i)}$ represent all symbols in $\underline{\boldsymbol{x}}$ except $\underline{\boldsymbol{x}}^i$. We use z_m to represent the m-th parity-check in the parity-check matrix and $\underline{\boldsymbol{z}}$ to stand for all parity-checks related to the symbol subblock $\underline{\boldsymbol{x}}$. In addition, $\underline{\boldsymbol{z}}^i$ denotes the parity-checks related to the code symbol $\underline{\boldsymbol{x}}^i$. In the nonbinary BP algorithm [38], $r_{m\to n}$ is the probability distribution sent from check m to symbol n, and $q_{n\to m}$ is the probability distribution sent from symbol n to check m. Particularly, $r_{m\to n}^a$ and $q_{n\to m}^a$ are the probabilities for $x_n = a$, with $a \in$ GF(q), in the distributions $r_{m\to n}$ and $q_{n\to m}$, respectively.

Since the nonbinary BP algorithm is a generalization of the binary case, it is sometimes called q-ary BP (QBP) decoding, which becomes binary BP with q=2. Similarly, the improved nonbinary BP decoding can be also called "improved q-ary BP (IQBP) decoding".

In the initialization step of the IQBP decoding, we set $q_{n\to m} = q_n^{(Ch)}$ for all n. Then the row step is identical to the row step of the QBP algorithm:

$$r_{m\to n}^{a} = \sum_{\left\{\boldsymbol{x}_1^{N_s} | x_n = a\right\}} P\left(z_m = 0 \mid \boldsymbol{x}_1^{N_s}\right) \prod_{j\in N(m)\backslash n} q_{j\to m}^{x_j}, \forall a \in \mathrm{GF}(q). \tag{4.6}$$

In this chapter, we implement the row step using a fast Fourier transform (FFT) [39], [40].

In the column step, the APP distribution of symbol n is computed in a similar way as in (4.2):

$$\begin{aligned} q_n &\triangleq P\left(x_n \mid \boldsymbol{y}_1^N, \underline{\boldsymbol{z}} = \underline{0}\right) \\ &= \frac{1}{P(\underline{\boldsymbol{x}})} \sum_{\underline{\boldsymbol{x}}^{(c')}} \left[P\left(\underline{\boldsymbol{x}} \mid \boldsymbol{y}_1^N\right) \prod_{i=0}^{p'-1} P\left(\underline{\boldsymbol{x}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0}\right) \right] \\ &= \frac{1}{P(\underline{\boldsymbol{x}})} P\left(x_n \mid \underline{\boldsymbol{z}}^{c'} = \underline{0}\right) \sum_{\underline{\boldsymbol{x}}^{(c')}} \left[P\left(\underline{\boldsymbol{x}} \mid \boldsymbol{y}_1^N\right) \prod_{i=0, i\neq c'}^{p'-1} P\left(\underline{\boldsymbol{x}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0}\right) \right], \end{aligned} \tag{4.7}$$

where $P(\underline{\boldsymbol{x}})$ is always assumed to be uniformly distributed; the distribution $P(x_n \mid \underline{\boldsymbol{z}}^{c'} = \underline{0})$ is calculated as

$$P\left(x_n = a \mid \underline{\boldsymbol{z}}^{c'} = \underline{0}\right) = \alpha_n \prod_{j\in N(n)} r_{j\to n}^{a}, \forall a \in \mathrm{GF}(q). \tag{4.8}$$

Similarly, $P(\underline{\boldsymbol{x}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0})$ is calculated as

$$P\left(\underline{\boldsymbol{x}}^i = a \mid \underline{\boldsymbol{z}}^i = \underline{0}\right) = \alpha_{(n-c'+i)} \prod_{j\in N(n-c'+i)} r_{j\to n-c'+i}^{a}, \forall a \in \mathrm{GF}(q). \tag{4.9}$$

Finally, the probability distribution sent from symbol n to check m is computed by

$$q_{n\to m}^{a} = \alpha_{nm}\, q_{n}^{a} \big/ r_{m\to n}^{a}, \forall a \in \mathrm{GF}(q). \tag{4.10}$$

Note that, in (4.8) - (4.10), α_n, $\alpha_{n\text{-}c'+1}$ and α_{nm} are normalization factors. Equation (4.10) is conceptually correct, but divided-by-zero errors may occur during the computation of $q_{n\to m}$. In our implementation, we actually calculate the distribution $q_{n\to m}$ by an alternate way, which is

$$\begin{aligned} q_{n\to m} = \alpha_{nm} \frac{1}{P(\underline{\boldsymbol{x}})} \left(\prod_{j\in N(n)\setminus m} r_{j\to n} \right) \\ \cdot \sum_{\underline{\boldsymbol{x}}^{(c')}} \left[P\left(\underline{\boldsymbol{x}} \mid \boldsymbol{y}_1^N\right) \prod_{i=0, i\neq c'}^{p'-1} P\left(\underline{\boldsymbol{x}}^i \mid \underline{\boldsymbol{z}}^i = \underline{0}\right) \right], \end{aligned} \tag{4.11}$$

where the summation part is exactly the same as the one in (4.7) and does not need to be re-computed.

4.4.2 Channel detector for IQBP

Since the OBBD cannot work with the IQBP decoder, we need to modify the original OBBD to get a new detector. On the one hand, the APP distributions $q_n^{(Ch)}$ need to be generated by the channel detector. We can manage the OBBD in the same way as in [53], where the probability distributions of non-overlapped p-bit-long subblocks are computed by the detector. On the other hand, we also need $P(\underline{\boldsymbol{x}} \mid \boldsymbol{y}_1^N)$ in (4.7), which actually are the APP distributions of overlapped ($p \cdot p'$)-bit-long subblocks, where the sliding window moves forward in p-bit (one code symbol) steps. Meanwhile, the channel detector will not output the bit LLRs, because they are not needed in the IQBP algorithm. These modifications of the OBBD are straightforward; for any particular subblock, the probability distribution can still be computed by (3.1), (3.2) and (3.3).

4.4.3 Boundary Management

At block boundaries, we manage the channel detector and IQBP decoder in a similar way as in Section 4.1. The channel detector generates probabilities for superposed subblocks covering from ac' to p' code symbols on the left boundary, and covering p' to c' code symbols on the right boundary. The IQBP decoder is also modified to take fewer channel messages into account at the block boundaries, during the computation of (4.7).

4.5 Nonbinary LDPC coded PMRCs

In this section, we evaluate the performance of IQBP decoding on nonbinary LDPC coded PMRCs, and compare the performance of IBP with conventional nonbinary BP decoding, where we still run the channel detector only once for each sector and perform at most fifty QBP or IQBP iterations.

We design a nonbinary LDPC code using the PEG construction method; the null space of a 114 by 1138 parity-check matrix over GF(16), with a constant column weight of three, gives a (1138, 1024) nonbinary LDPC code. Each nonbinary LDPC codeword encodes 4096 user bits, as in the binary LDPC code in Section 4.2, and both codes have similar code rate.

Before we simulate the IQBP decoder on PMRCs, the decoder parameters p' and c' need to be carefully determined by analyzing the dependence between channel messages, as we did in Sections 4.2 and 4.3. However, the channel messages for IQBP decoding are probability distributions, and their correlations cannot be measured by a simple autocorrelation sequence. Fortunately, the channels investigated in this paper have binary inputs. No matter how the channel is encoded, the bit LLRs can always be utilized to

investigate the correlations among channel messages. Keeping in mind that a code symbol corresponds to p consecutive bits, if an LLR is strongly dependent on the l LLRs immediately before it and the l LLRs immediately after it, then a code symbol is also strongly dependent on the l LLRs before the symbol and the l LLRs after the symbol. Therefore, $c' = ac' = \lceil l/p \rceil$ is enough for the IQBP decoder to cover all significantly dependent channel messages.

As in Section 4.3, we consider two nonbinary LDPC coded PMRCs with $D_u = 0.8741$ and $D_u = 1.2238$, respectively. From Fig. 4.6, we find that $l = 4$ is enough to cover the most significant LLRs for both PMRCs. Given that the nonbinary LDPC code is over GF(16), i.e., p=4, $c' = ac' = 1$ is sufficient for the IQBP decoder. Actually, for complexity reasons, p'=3 is the largest value we can use in our simulations. We show the simulation results in Figs. 4.11 and 4.12, and observe additional gains. Again, it is clear that the small gain for the low density channel is due to the weaker dependence between channel messages. For the high density channel, the gain over QBP is larger than 0.1 dB, which is a small gain but still significant, because it is very difficult to get gains over nonbinary LDPC coded and QBP decoded PMRCs. In addition, the performance of binary LDPC coded PMRCs with IBP decoders is very close to that of nonbinary LDPC coded and QBP decoded PMRCs. This shows that, for perpendicular magnetic recording, the error correction capability of binary LDPC codes is comparable to that of nonbinary LDPC codes.

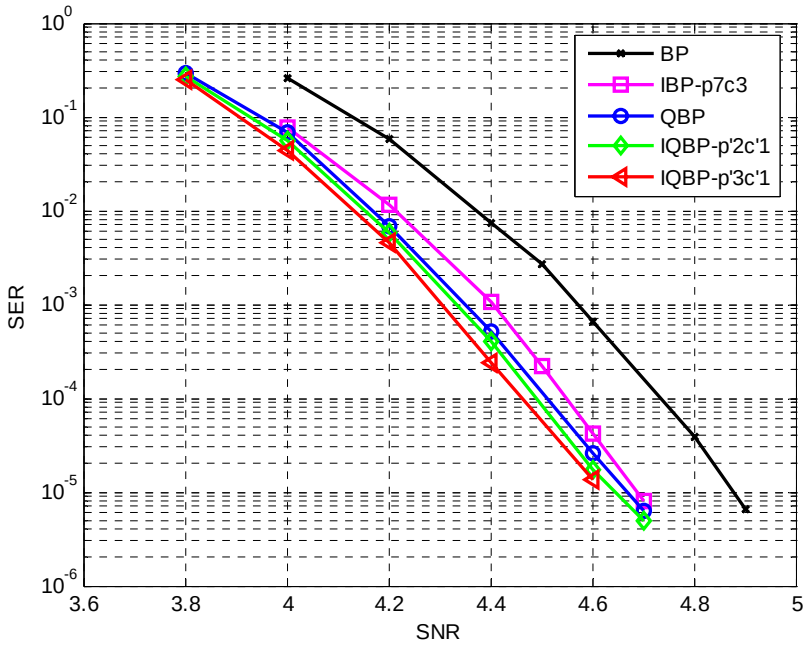


Fig. 4.11. Performance of nonbinary LDPC coded and IQBP decoded PMRCs with D_u = 0.8741.

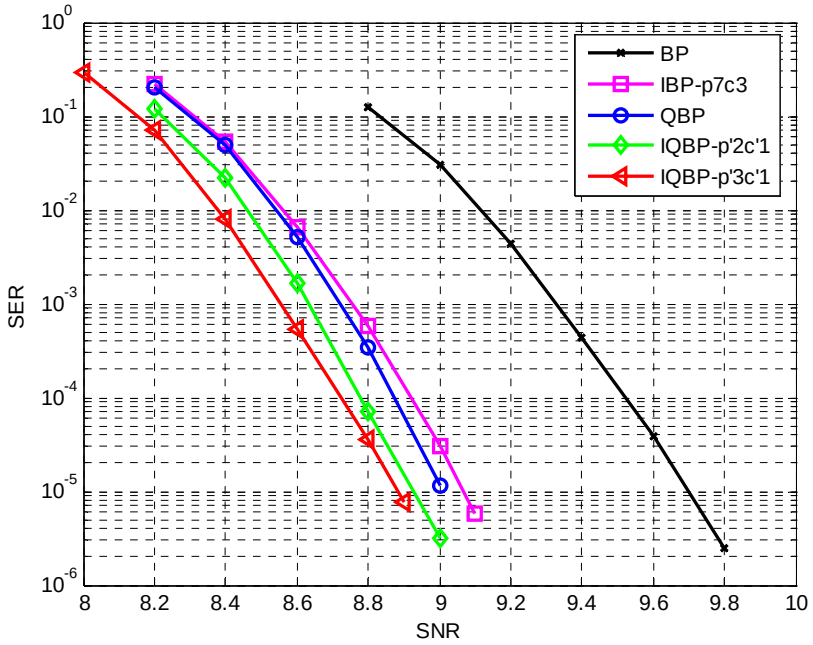


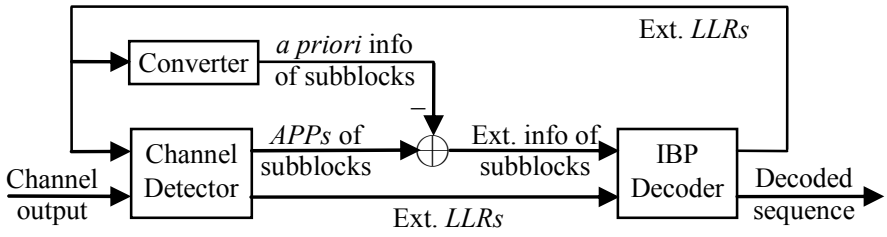
Fig. 4.12. Performance of nonbinary LDPC coded and IQBP decoded PMRCs with D_u = 1.2238.

4.6 Turbo equalization

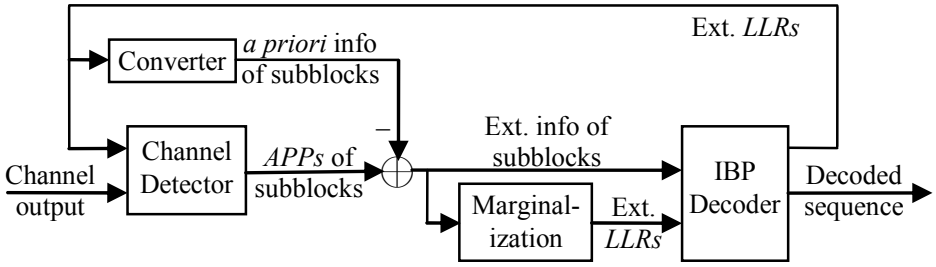
So far, we have shown that the improved BP decoding outperforms the original BP decoding when the channel detector is used only once. However, it is well known that the decoding performance of LDPC coded channels can be improved by turbo equalization, which performs multiple channel iterations (or turbo iterations) by feeding the extrinsic information generated by the LDPC decoder back to the input of the channel detector. It is interesting to compare the performance of BP and IBP algorithms with turbo equalization. In this chapter, we will limit the investigation of turbo equalization to binary LDPC coded channels.

For the original BP decoding, the implementation of turbo equalization is easy. Since the messages transferred between the BCJR channel detector and BP decoder are LLRs for bits, the extrinsic information is obtained simply by subtracting the *a priori* LLRs from the *a posteriori* LLRs. However, for IBP decoding, it is a bit more complicated. The OBBD channel detector takes in LLRs for bits, but outputs both bit-LLRs and probability distributions for overlapped subblocks. The extrinsic bit-LLRs could be computed as usual; but we need to carefully consider the calculation of the extrinsic information for subblocks. A straightforward way is subtracting (in the logarithm domain) the *a priori* distributions of subblocks from the *a posteriori* distributions, where the *a priori* distributions of subblocks could be computed from the LLRs at the input of the OBBD. We show the diagram of such a turbo equalization method in Fig. 4.13(a).

This equalization method looks good but it has a subtle problem. Due to the nature of the detection on the trellis, the OBBD generates consistent soft information, i.e., the bit probabilities are equal to the probabilities marginalized from the distributions of



(a) Compute extrinsic information simply by subtraction in the logarithm domain.



(b) Compute extrinsic information for bits by marginalization.

Fig. 4.13. Skewed turbo equalizations for IBP decoding.

corresponding subblocks. In other words, the APPs of the channel detector are always consistent. However, when we subtract the *a priori* information from the APPs as in Fig. 4.13 (a), the consistency is broken; the extrinsic information for bits is different from that marginalized from the extrinsic information of subblocks. The theoretical proof of this inconsistency is straightforward, and we have a skewed turbo equalization method.

One may want to solve the inconsistency problem in such a way that the extrinsic information for bits is marginalized from the extrinsic information of subblocks, while the bit LLRs given by OBBD are discarded, as shown in Fig. 4.13(b). However, since the subblocks are overlapped, it is easy to prove that soft information on a given bit, marginalized from different subblocks will not necessarily be the same. Therefore, this turbo equalization is still a skewed one and more complicated than the turbo equalization in

Fig. 4.13(a) due to the extra marginalization step. We will use the skewed turbo equalization in Fig. 4.13(a), even though we cannot measure the impact of the inconsistency.

A couple of iterative schemes for both BP and IBP decoding are tested on the PMRC with $D_u = 1.2238$. For the standard BP decoding, we use ten BP (local) iterations, which is a typical choice, but we find that more turbo (global) iterations continually improve performance. We show the SER for BP decoding with at most 6, 8 and 10 turbo iterations in Fig. 4.14, where we achieve a gain of more than 0.5 dB over BP decoding with at most 50 BP iterations, but without turbo equalization. We observe that more than ten turbo iterations give further but marginal performance improvement. For IBP decoding, the reference curve is the one for p=5 and c=2, with fifty IBP (local) iterations and without turbo equalization, which we have shown in Section 4.3. The performance with at most ten turbo iterations and ten local iterations is worse than IBP decoding without turbo equalization, which means that the turbo iterations are not as helpful as in standard BP decoding. We then increase the number of local iterations from ten to fifty, and test the performance with at most three and five turbo iterations. From Fig. 4.14, we see that these two iterative schemes have almost the same performance, and give only a small gain over the reference IBP decoder. It is clear that the (skewed) turbo equalization for IBP works, but it cannot provide significant additional gains. An interesting observation is that standard BP decoding with turbo equalization achieves a comparable performance to IBP decoding.

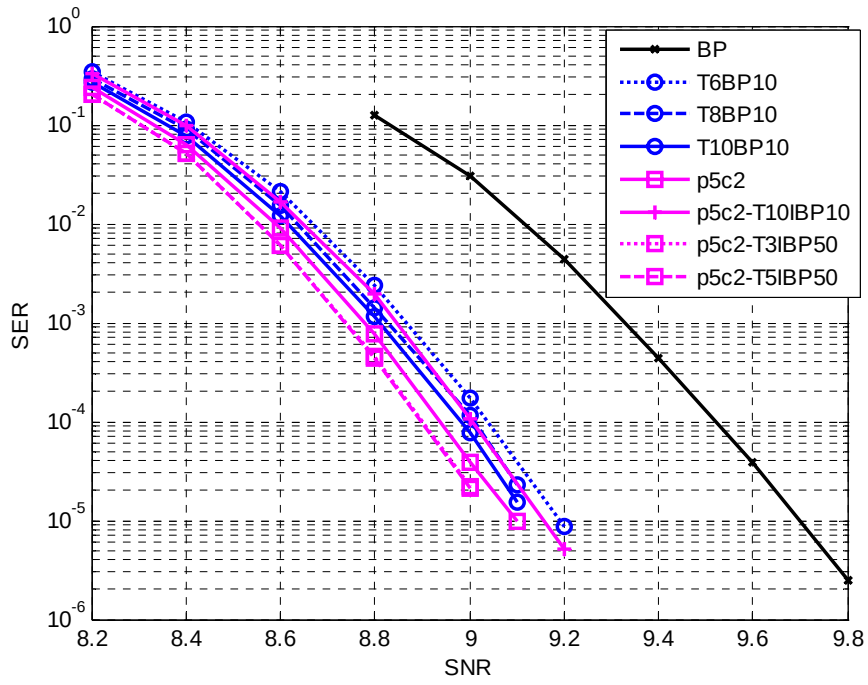


Fig. 4.14. Performance of turbo equalized BP and IBP decoding on a PMRC with $D_u = 1.2238$. Note that the curves for p5c2-T3IBP50 and p5c2-T5IBP50 almost overlap and are not distinguishable.

We also implement the turbo equalization for a PMRC with $D_u = 0.8471$, and present the results in Fig. 4.15, where the turbo equalization provides a small performance improvement for both BP and IBP decoding.

Given two methods with comparable performance, the choice will hinge on complexity. However, since soft iterative decoding is a dynamic process (the actual number of iterations is random), and different implementations of the same algorithm will give distinct complexities, we will not provide an operation count for the algorithms. Instead, we use a simple analysis to highlight the complexity difference between BP and IBP decoding. The standard BCJR algorithm and the OBBD perform the same computation of the channel metric, forward and backward recursions, and the output LLRs for bits. But the

OBBD needs to calculate the probability distributions for overlapped subblocks, whose complexity increases exponentially with the subblock length p, in terms of both computational time (time complexity) and memory usage (space complexity). Similarly, the IBP decoding algorithm does the same check-to-bit step as in BP, but a far more complicated bit-to-check step, whose time and space complexities also increase exponentially with the subblock length p. For standard BP, additional turbo iterations may increase the time complexity but it still has an advantage in terms of space complexity over IBP decoding.

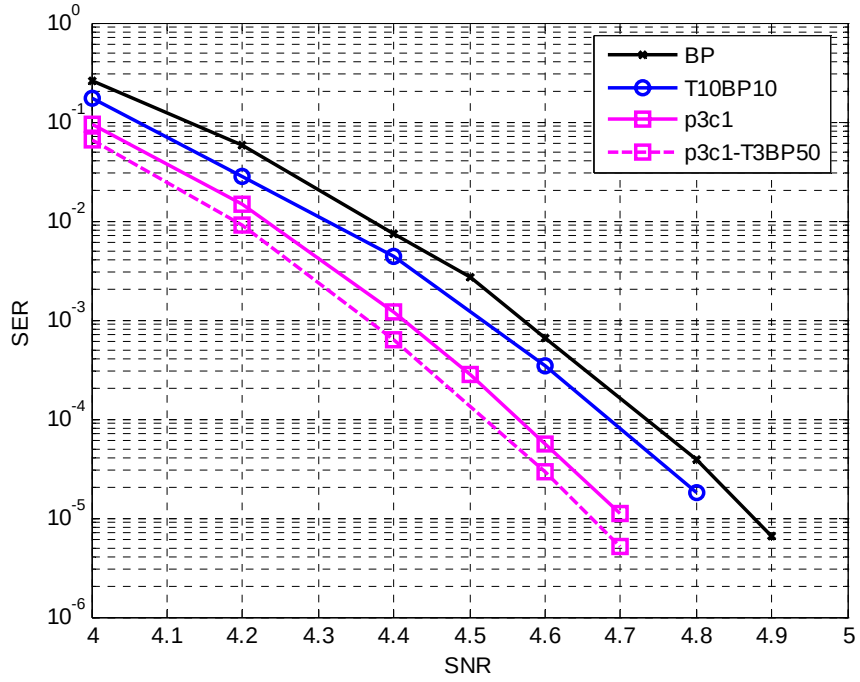


Fig. 4.15. Performance of turbo equalizations of BP and IBP decoding on the PMRC with $D_u = 0.8471$.

Finally, we would like to give some quick guidance on the time complexity of BP and IBP decoding. All of our simulations run on the same type of platform. By measuring the running speed (blocks per second) of BP and IBP on a PMRC with $D_u = 1.2238$, we find

that, at SNR = 9.1dB, the BP decoded channel with at most ten turbo iterations (T10BP10) is 35% faster than IBP with p=5 and c=2, but without turbo equalization. As another example, note that from Figs. 4.10 and 4.14, the performance of IBP decoding with p=3 and c=1 is slightly better than BP decoding with at most six turbo iterations (T6BP10). At SNR = 9.2dB, we observe that the IBP decoded channel is 11% faster.

4.7 Discussion

Without turbo equalization, IBP decoding exhibits significant performance gains over the standard BP decoding. Our channel detectors are implemented using BCJR-based forward and backward algorithms, but other detectors with lower complexity could be considered, such as the SOVA and the forward maximum *a posteriori* probability (forward-MAP) algorithms proposed in [55]. The IBP decoders are capable of providing gains over the standard BP decoder even with these simpler channel detectors, given that the channel messages generated by these detectors are always correlated. In addition, the IBP decoding is also expected to work with noise-predictive detectors [56], [57], although we did not include it in this work. The noise-prediction whitens the noise at the receiver, but the channel messages are still severely correlated. The investigation on the EPR4 channel with AWGN in Section 4.2 backs up this assertion.

Turbo iterations substantially improve the performance of BP decoding making it comparable to IBP decoding, but do not help significantly the IBP decoder. Generally speaking, turbo equalization is a method to iteratively get more information from the channel detector for the benefit of the LDPC decoder. Since, with turbo equalization, IBP and BP decoding exhibit comparable performance, it appears that the role of the turbo

iterations might be to extract information about the correlations among channel messages. If this is the case, then the IBP decoder provides an alternative way of harvesting the performance gain, which was previously done by turbo equalization of the standard BP decoder.

5 Constructing LDPC Codes for Magnetic Recording with Fewer Short Cycles

BP decoding of LDPC codes has been shown to provide excellent performance on a wide variety of channels. Although the presence of cycles on the factor graphs [43] of LDPC codes makes the BP decoder sub-optimal, it is believed that shorter cycles are more harmful than longer ones. The shortest cycles defined on the graphs of LDPC codes are of length four, hereby referred to as cycle-4's, which have been avoided in all known code construction techniques. Furthermore, since the largest number of independent BP iterations is limited by the girth of the LDPC code [34], [50], it is desirable to construct LDPC codes with girths as large as possible [50], [58], not just free of cycle-4's, where the girth of an LDPC code is defined as the length of the shortest cycle on its graph.

The length of LDPC codes designed for magnetic recording matches the sector size, e.g., 512 bytes, and their rate is very high, around 0.9, to avoid severe channel density penalty, which means that the size of the parity-check matrix **H** of the code is essentially fixed. Given that the column weight of **H** is not smaller than three, the largest girth that can be achieved is also determined. If we use either the PEG algorithm [50] or the integer lattice construction [58] to design LDPC codes for magnetic recording, the largest girth we can get is only six. We also note that so far there are no reports of girth-eight LDPC codes in the literature for the set of parameters of interest. However, to further improve the design of LDPC codes for magnetic recording, we would like to reduce the number of the shortest cycles (cycle-6's) during code construction. It has been shown in [59] and [60] that the dominant trapping sets of an LDPC code are closely related to the shortest cycles.

Therefore, reducing the number of shortest cycles may eliminate some dominant trapping sets and improve the decoding performance of LDPC codes, especially at high SNR.

In this chapter, we introduce some methods to minimize the number of shortest cycles during LDPC code construction, and then verify the performance of the constructed LDPC codes on PMRCs. We consider both random and deterministic constructions of LDPC codes. In Section 5.1, we discuss a simple modification of the PEG algorithm to reduce the number of short cycles. In Section 5.2, we apply the modified PEG (MPEG) algorithm to the random construction of quasi-cyclic (QC) LDPC codes. We present the simulation results of the constructed LDPC codes on PMRCs in Section 5.3. We introduce a deterministic code construction technique based on a rectangular integer lattice and evaluate the performance of the LDPC codes designed in Section 5.4. Finally, we provide a brief discussion of the proposed code construction techniques and draw some conclusions in Section 5.5.

5.1 The modified PEG algorithm

In this section, we briefly review the PEG algorithm presented in [50] and then introduce a simple modification that leads to the construction of LDPC codes with fewer shortest cycles. On the bipartite graph of an LDPC code, let V_c denote the set of all check nodes, V_s the set of all symbol nodes and E the set of all edges. For a given symbol node s_j, a tree could be expanded to a depth of l. Define $\mathcal{N}_{s_j}^l$ as the set consisting of all check nodes reached by this tree and $\bar{\mathcal{N}}_{s_j}^l$ as its complementary set, i.e., $\bar{\mathcal{N}}_{s_j}^l = V_c \setminus \mathcal{N}_{s_j}^l$. The PEG algorithm constructs the graph of an LDPC code one edge at a time. When a new edge is added to a symbol node, the algorithm maximizes the local girth of the node.

We repeat here the pseudo-program of the PEG algorithm given in [50] for constructing a bipartite graph with m check nodes and n symbol nodes, where d_{s_j} is the degree of symbol node s_j.

Progressive Edge-Growth (PEG) Algorithm [50]
for j =0 **to** n – 1 **do**
begin
  for k = 0 **to** d_{s_j}– 1 **do**
  begin
    if $k = 0$
    $E_{s_j}^0 \leftarrow$ edge (c_i, s_j), where $E_{s_j}^0$ is the first edge incident to s_j, and c_i is a check node having the lowest check degree under the current graph setting $E_{s_0} \bigcup E_{s1} \bigcup \cdots \bigcup E_{s_{j-1}}$.
   else
    expanding a tree from symbol node s_j up to depth l under the current graph setting such that the cardinality of $\mathcal{N}_{s_j}^l$ stops increasing but is less than m, or $\bar{\mathcal{N}}_{s_j}^l \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$, then $E_{s_j}^k \leftarrow$ edge (c_i, s_j), where $E_{s_j}^k$ is the k-th edge incident to s_j and c_i is one check node picked from the set $\bar{\mathcal{N}}_{s_j}^l$ having the lowest check node degree.
  end
end

In the PEG construction, when multiple check nodes with the same lowest degree are available in $\bar{\mathcal{N}}_{s_j}^l$, one of them is randomly chosen to connect to the symbol node s_j. To reduce the number of short cycles, a simple modification is made for cases where $k > 0$ and $\bar{\mathcal{N}}_{s_j}^l \neq \varnothing$ but $\bar{\mathcal{N}}_{s_j}^{l+1} = \varnothing$. For these cases, the number of occurrences of each check node in $\bar{\mathcal{N}}_{s_j}^l$ at the depth l + 1 of the tree is recorded and denoted as the multiplicity M_i. Connecting a check node in $\bar{\mathcal{N}}_{s_j}^l$ to s_j will generate exactly M_i short cycles with length 2(l + 2). In the proposed modification, when multiple check nodes having the same lowest degree are available in $\bar{\mathcal{N}}_{s_j}^l$, those with the lowest multiplicity

are preferred. In this situation, if we still have more than one choice, one of them will be randomly selected.

Using both the PEG algorithm and the MPEG algorithm proposed in this section, we design LDPC codes for magnetic recording, where their parity check matrices have 456 rows and 4560 columns with constant column weight of three. For each algorithm, we perform ten trials and select the code with the smallest number of shortest cycles. We obtain a girth-six PEG-LDPC code with 24445 shortest cycles (cycle-6's), and a girth-six MPEG-LDPC code with 10617 cycle-6's. Apparently, this simple modification on the PEG algorithm is very effective in reducing short cycles in this example.

In addition, we attempt to further improve the MPEG algorithm by applying a similar modification to the look-ahead-enhanced version of the PEG algorithm in [50]. In the look-ahead-enhanced version of the MPEG algorithm, we follow the MPEG algorithm as usual, except when there are several choices for placing the *k*-th edge. For each candidate check nodes, the largest depth *l* and the smallest multiplicity of check nodes in $\bar{\mathcal{N}}_{s_j}^{l}$ are evaluated on the tree expanded from s_j given that an edge is temporarily put on the graph to connect the candidate check node with s_j. Among the candidate check nodes having the same largest depth *l*, we randomly pick one with the smallest multiplicity. This enhanced version of the MPEG algorithm tries much harder to find better LDPC codes. However, with our design parameters (the size and the column weight of **H**), it cannot construct an LDPC code with fewer than 10617 cycle-6's after ten trials. This enhancement of the MPEG algorithm does not lead to any obvious improvement in the design of LDPC codes for magnetic recording, although it may work better for

constructing LDPC codes with lower rates and sparser parity-check matrices.

5.2 Constructing quasi-cyclic LDPC codes with the MPEG algorithm

QC-LDPC codes have low encoding complexity as well as low decoding complexity. In this section we present a design method, which solely aims to reduce the number of short cycles during code construction. In other words, we are interested in the minimum number of short cycles, which can be achieved in light of the quasi-cyclic structure.

5.2.1 Cycles in QC-LDPC codes

A QC-LDPC code has a special parity-check matrix consisting of small square blocks, which are the zero matrix or circulant permutation matrices. The circulant weight of the permutation matrix is defined as the column weight (or equivalently the row weight) of the permutation matrix. Our discussion is limited to QC-LDPC codes, which have permutation matrices with unit circulant weight. Let P^a with $0 \le a < L$ be the circulant permutation matrix obtained by shifting the $L \times L$ identity matrix $\mathbf{I}$ to the right a times. To simplify the notation, the $L \times L$ zero matrix is denoted by P^{∞}. Then an $mL \times nL$ parity-check matrix of the QC-LDPC code can be expressed as

$$\mathbf{H} = \begin{pmatrix} P^{a_{11}} & \cdots & P^{a_{1n}} \\ \vdots & \ddots & \vdots \\ P^{a_{m1}} & \cdots & P^{a_{mn}} \end{pmatrix}. \tag{5.1}$$

Depending on the values of the a_{ij}'s in (5.1), the QC-LDPC code could be regular or irregular. Especially, when all a_{ij}'s take finite values from $\{0, 1, \ldots, L\text{-}1\}$, the $\mathbf{H}$ represents an (m, n)-regular LDPC code, whose rank is no more than $mL - m + 1$, and whose girth is at most twelve for any $m \ge 2$ and $n \ge 3$. There are many interesting properties of

QC-LDPC codes investigated in [61]-[64], which readers are referred to for more details. Here, we only introduce one important cycle property of QC- LDPC codes, which will be used in our code design.

We can construct an $m \times n$ matrix $\mathbf{M}(\mathbf{H})$ for the $\mathbf{H}$ in (5.1), which is called the mother matrix of $\mathbf{H}$ in [64], by substituting "0" for each $L \times L$ zero matrix and "1" for each $L \times L$ circulant permutation matrix. Then the cycles in $\mathbf{M}(\mathbf{H})$ are identifying the block-cycles in $\mathbf{H}$. The block-cycles do not necessarily generate cycles in $\mathbf{H}$, but cycles in $\mathbf{H}$ must be caused by block-cycles. Therefore, the number of shortest cycles in $\mathbf{H}$ is a multiple of L. A block-cycle with a length of $2l$ can be expressed by the chain $P^{a_1} \rightarrow P^{a_2} \rightarrow \ \ldots \ \rightarrow P^{a_{2l}} \rightarrow P^{a_1}$, which will lead to cycles of length $2lb$, if b is the least positive number such that

$$b \cdot \sum_{i=1}^{2l} (-1)^{i-1} a_i = 0 \ \bmod L \,. \tag{5.2}$$

The proof of (5.2) is given in Proposition 3 in [64].

5.2.2 Constructing QC-LDPC codes with fewer short cycles

To design QC-LDPC codes with fewer short cycles, we apply the MPEG algorithm to the random construction of QC-LDPC codes. For convenience, $\mathbf{S}(\mathbf{H})$, a simplified representation of $\mathbf{H}$, is defined in (5.3) and will be utilized in our code design.

$$\mathbf{S}(\mathbf{H}) = \begin{pmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \cdots & a_{mn} \end{pmatrix}. \tag{5.3}$$

Given the values of L, m, n and the desired girth g, the procedure for the construction of a QC-LDPC code with the MPEG algorithm is hereby referred to as Algorithm 5.1, and

described as follows.

Algorithm 5.1

1) Construct the $m \times n$ mother matrix $\mathbf{M}(\mathbf{H})$ using the MPEG algorithm to reduce the number of short block-cycles.

2) Initialize the $m \times n$ matrix $\mathbf{S}(\mathbf{H})$ with infinities "∞". Then for each location of a "1" in $\mathbf{M}(\mathbf{H})$, a finite value $a_i \in \{0, 1, \ldots, L-1\}$ is assigned to the same location in $\mathbf{S}(\mathbf{H})$ in the following way.

for $i = 1$ **to** n **do**
begin
  $k = 1$;
  for $j = 1$ **to** m **do**
  begin
    if $\mathbf{M}(\mathbf{H})\ (i, j) = 1$
      if $k = 1$
        $\mathbf{S}(\mathbf{H})\ (i, j) \leftarrow a$, which is randomly chosen from $\{0, 1, \ldots, L-1\}$
      else
        $\mathbf{S}(\mathbf{H})\ (i, j) \leftarrow a = 0$. Then list all block-cycles no longer than g, which are going through position (i, j) and formed by elements with finite values in $S\ (H)$. Vary the value of a from 0 to $L-1$ and evaluate these block-cycles using (5.2).
        if all values in $\{0, 1, \ldots, L-1\}$ lead to cycles shorter than g, the construction failed.
        else if there are values of a do not lead to any cycles in the list of block-cycles. Then $\mathbf{S}(\mathbf{H})\ (i, j) \leftarrow a$, which is randomly chosen from these values.
        else if there are values of a which only lead to cycles with length g in the list of block-cycles. Then $\mathbf{S}(\mathbf{H})\ (i, j) \leftarrow a$, which cause the smallest number of cycles of length g.
    $k = k + 1$**;**
  end
end

3) Construct the $mL \times nL$ parity-check matrix $\mathbf{H}$ according to $\mathbf{S}(\mathbf{H})$.

5.2.3 Constructing QC-LDPC codes for magnetic recording

For magnetic recording, the code rate should be around 0.9, while the information

encoded is no less than 4096 bits for standard size sectors. We choose $mL = 456$ and $nL = 4560$ to keep the information overhead as low as eight bits, while the overhead may be larger when **H** is not full rank. By varying L from one to 152, there are 14 combinations of L, m and n to satisfy the conditions, which are given in Table 5.1. For each combination, we design a QC-LDPC code with column weight three by performing ten trials of Algorithm 5.1, and the minimum number of shortest cycles (cycle-6's) is computed and also listed in Table 5.1.

Table 5.1 QC-LDPC codes for magnetic recording with column weight three

L	m	n	Total # of cycle-6's	Total # of cycle-8's
1	456	4560	10617	1577808
2	228	2280	15366	1571800
3	152	1520	17556	1544238
4	114	1140	18640	1525880
6	76	760	18510	1529658
8	57	570	17120	1554584
12	38	380	15924	1547292
19	24	240	14668	1543522
24	19	190	13944	1547784
38	12	120	12502	1558760
57	8	80	11742	1594689
76	6	60	10488	1581712
114	4	40	9462	1626552
152	3	30	7752	1693128

We can see that all constructions in Table 5.1 give LDPC codes with fewer cycle-6's than the code designed by the original PEG algorithm, which has 24445 cycle-6's. Note that the LDPC code constructed by the MPEG algorithm is treated as a special case of a QC-LDPC code with $L = 1$. In addition, when $L = 152$, $\mathbf{M}(\mathbf{H})$ is an all one matrix; the

MPEG construction in the first step of Algorithm 5.1 is trivial.

5.2.4 Simulations and discussion

The PMRC considered is equalized to GPR4 targets, while a mix of 90% jitter noise power and 10% electronics noise power is assumed in all simulations.

To evaluate the performance of the LDPC codes with different number of shortest cycles (cycle-6's), we simulate the PEG-LDPC code designed in Section 5.1, which has 24445 cycle-6's, and four QC-LDPC codes constructed by Algorithm 5.1 with L = 1, 4, 38 and 152, which are highlighted in Table 5.1. Note that the LDPC codes with L = 38 and 152 are strictly regular, while other codes are approximately regular, i.e., only of constant column weight three. In addition, all parity-check matrices of these LDPC codes have full rank, except the one with L =152, which has a rank of 454, that is the theoretical upper limit for this case. In other words, the LDPC code with L = 152 has a code rate higher than 0.9, while the rate of other codes is exactly 0.9. Usually, a higher code rate will lead to a smaller channel density penalty in a MRC. However, we always assume that there are only 4096 user information bits, while all other (overhead) information bits are dummy bits. Therefore, for fair comparison, all LDPC codes (which have the same length) are simulated at the same channel density.

Shown in Fig. 5.1 are the simulation results at D_c = 0.9713 with at most 50 BP iterations of LDPC decoding, where the LDPC codes with fewer cycle-6's exhibit better performance. The explanation for this illustrative result is two-fold. On one hand, the dominant trapping sets, which are making the major contribution at high SNR, are closely related to the short cycles in the LDPC code [59], [60]. On the other hand, the SNRs simulated are high enough, since all performance curves tend to have relatively flat tails,

i.e., they are going into the error floor.

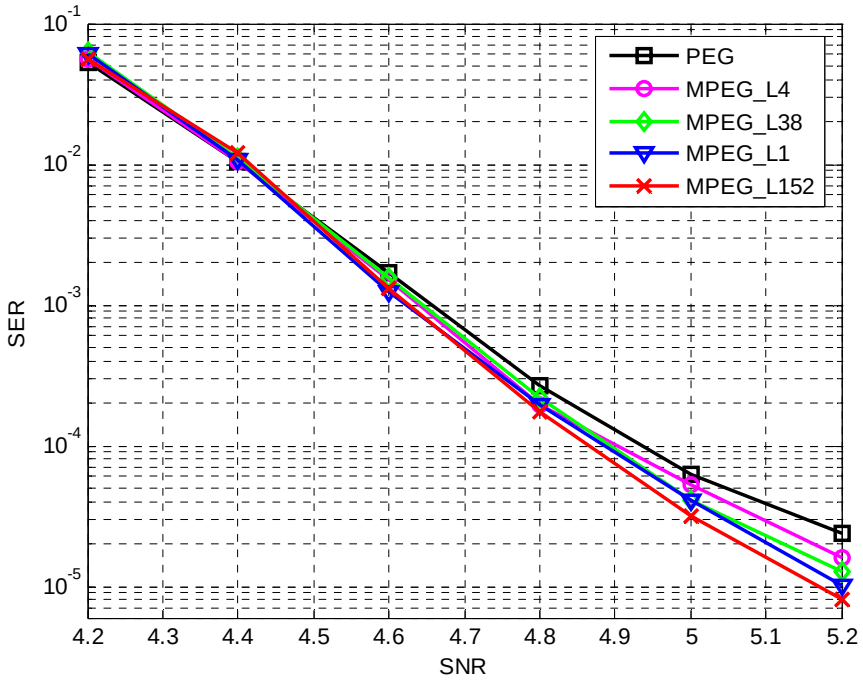


Fig. 5.1. Performance of LDPC codes with different number of cycle-6's at channel density 0.9713.

However, the performance pattern changes with increased channel density. As shown in Fig. 5.2, the LDPC codes with fewer cycle-6's do not necessarily provide better performance at channel density 1.3598. Higher channel density leads to stronger correlations between bits and hence cycles larger than six may contribute more to the decoding performance in the simulations. Therefore, we compute the number of cycle-8's for all LDPC codes we designed, while the PEG-LDPC code has 1457622 cycle-8's and the number of cycle-8's for all other LDPC codes is listed in Table 5.1. Clearly, when we minimize the number of cycle-6's using Algorithm 5.1, the number of cycle-8's is boosted. (Although there are a few small deviations, the general trend is clear.) In particular, the LDPC code with $L = 152$ has a very small number of cycle-6's but many more cycle-8's

than other codes. In this work, we did not have enough computational power to check the number of cycles longer than eight. But note that Step 2 of Algorithm 5.1 does not take care of cycles longer than the girth of the code being designed, while the MPEG algorithm is supposed to reduce the number of short cycles in a more general sense. Therefore, this could be the reason that the LDPC code constructed by the MPEG algorithm (L = 1) provides the best performance in this case.

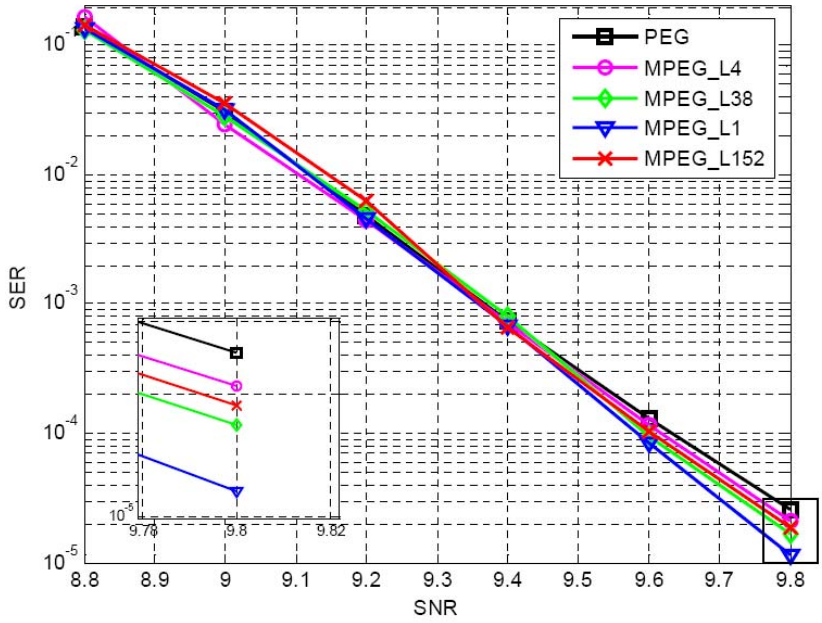


Fig. 5.2. Performance of LDPC codes with different number of cycle-6's at channel density 1.3598.

So far, our simulations illustrate very well the effect of short cycles in LDPC codes on PMRCs. The MPEG algorithm and Algorithm 5.1 can also be used to construct LDPC codes with higher column weight. However, although we can easily design LDPC codes with column weight four, we will have difficulty verifying the effect of short cycles, because we cannot simulate the codes at high enough SNRs. In other words, it is difficult to

reach the error floor of LDPC codes with column weight of four by simulation. But it is reasonable to expect that reducing the number of short cycles by the MPEG algorithm and Algorithm 5.1 could be always helpful.

One thing we need to point out is that the number of short cycles may not be the only factor affecting the performance of LDPC codes. Identifying the structural features, which determine the performance of LDPC codes, is still an open problem in error correcting coding. We purposefully kept a randomness element in all code constructions presented and we believe that this helps the MPEG algorithm and Algorithm 5.1 translate fewer short cycles into a performance gain without much change in other structural features of the LDPC codes.

5.3 Lattice construction of QC-LDPC codes

In this section, we investigate a deterministic construction technique for LDPC codes for magnetic recording based on rectangular integer lattices [58], while reducing the number of cycle-6's. The lattice construction method proposed in [58] generates QC-LDPC codes with prime L's. After a brief review of this technique, we indentify the cycle-6's on the lattice and generalize the construction method to the case where L is not a prime and then introduce a method to construct lattice-LDPC codes for magnetic recording with fewer cycle-6's.

5.3.1 Lattice construction of LDPC codes

Shown in Fig. 5.3 is a rectangular integer lattice $LA = \{(x, y)\colon 0 \le x \le K - 1, 0 \le y \le L -$

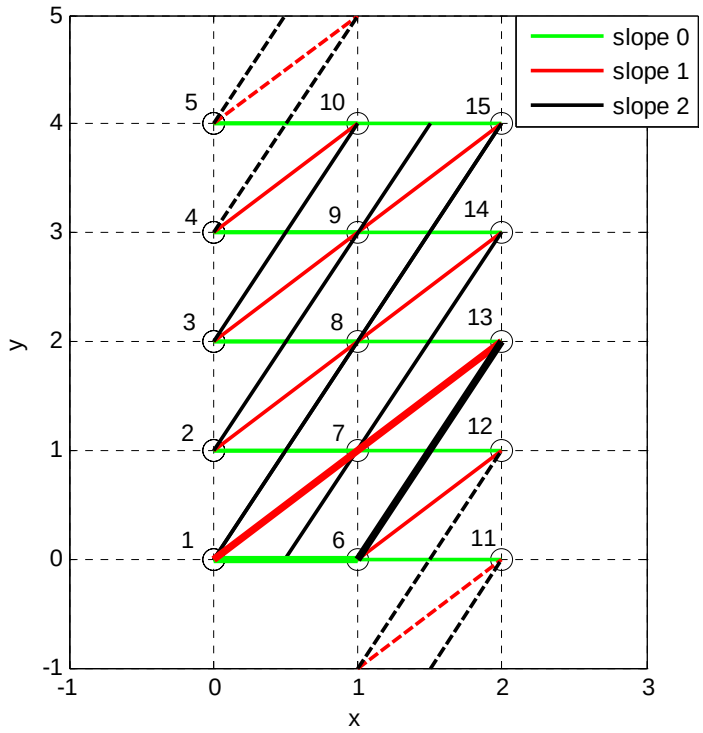


Fig. 5.3. A rectangular integer lattice with L=5 and K=3, where lines with slopes 0, 1 and 2 are depicted and a triangle is highlighted.

1} for $K = 3$ and $L = 5$, where $0 < K \leq L$ and L is a prime. The points on the lattice LA are labeled by a one-to-one mapping function $l(x, y)$. For easy understanding, we use the same mapping function as in [58], where $l(x, y) = L \cdot x + y + 1$. A set of K points $\{(x, a + sx \bmod L): 0 \leq x \leq K - 1\}$ for any fixed a in $\{0, \ldots, L - 1\}$ is called a line with slope s, where $0 \leq s \leq L - 1$. For a given L and any $0 < K \leq L$, if two lines have no common points, they are referred to as parallel. On a lattice, there are exact L lines for each slope and they are parallel. The set of points on parallel lines of slope s can be mapped on to an incidence matrix $\mathbf{H}_s$ of dimension $KL \times L$. Then the $KL \times L^2$ matrix $\mathbf{H} = [\mathbf{H}_0\ \mathbf{H}_1, \ldots, \mathbf{H}_{L-1}]$ defines a (K, L)-regular QC-LDPC code. The binary image of the $\mathbf{H}$ matrix for the lattice in Fig. 5.3 is given in [65] and [66]; we will not repeat it here. Instead, a more intuitive expression of

H is

$$\mathbf{H} = \begin{bmatrix} p^{-s_0 x_0} & p^{-s_1 x_0} & \cdots & p^{-s_{L-1} x_0} \\ p^{-s_0 x_1} & p^{-s_1 x_1} & \cdots & p^{-s_{L-1} x_1} \\ \vdots & \ddots & & \vdots \\ p^{-s_0 x_{K-1}} & \cdots & & p^{-s_{L-1} x_{K-1}} \end{bmatrix}, \tag{5.4}$$

where P^{-a} with $0 \le a < L$ is the circulant permutation matrix obtained by shifting the $L \times L$ identity matrix **I** to the left a times. This definition of P^{-a} is consistent with the one for P^{a} in Section 5.2 and hence it is still valid to judge block cycles using (5.2). Each block column in **H** is related to a slope while each block row is related to a value on the x-axis of the lattice.

For a given column weight K, to design a parity-check matrix **H** given by (5.4), one needs to find a set of slopes, which generate the smallest number of shortest cycles. It is extremely hard to design high rate LDPC codes this way. Instead, we consider a revision of this lattice construction proposed in [65], where the transpose of a sub-matrix of **H** in (5.4) is used to define a (M, K)-regular QC-LDPC code, where $0 < M \le L$ is the number of slopes selected, and is given by

$$\mathbf{H} = \begin{bmatrix} p^{-s_0 x_0} & p^{-s_0 x_1} & \cdots & p^{-s_0 x_{K-1}} \\ p^{-s_1 x_0} & p^{-s_1 x_1} & \cdots & p^{-s_1 x_{K-1}} \\ \vdots & \ddots & & \vdots \\ p^{-s_{M-1} x_0} & \cdots & & p^{-s_{M-1} x_{K-1}} \end{bmatrix}. \tag{5.5}$$

During the design of a code for magnetic recording, to find the set of M slopes, which gives the smallest number of cycle-6's, requires that all such cycles be identified on the lattice.

Proposition 5.1: Let **H** be the incidence matrix of lines on a rectangular integer lattice *LA*

$= \{(x, y): 0 \le x \le K - 1, 0 \le y \le L - 1\}$. Then each cycle-6 in **H** can be mapped to a unique triangle on the lattice.

Proof: Without loss of generality, we assume the incidence matrix **H** is of the form in (5.5). As discussed in Section 5.2, all cycles in an **H** with such structure come from their related block cycles. By picking up an arbitrary nonzero element from each of the six blocks which form a block cycle of length six, we have six points on the lattice: $(x_1, a_1 + s_1x_1 \bmod L)$, $(x_1, b_1 + s_2x_1 \bmod L)$, $(x_2, b_2 + s_2x_2 \bmod L)$, $(x_2, c_1 + s_3x_2 \bmod L)$, $(x_3, c_1 + s_3x_3 \bmod L)$ and $(x_3, a_2 + s_1x_3 \bmod L)$. To form a cycle-6, two elements with the same x_i must be in the same column, i.e., $a_1 + s_1x_1 = b_1 + s_2x_1 \bmod L$, $b_2 + s_2x_2 = c_1 + s_3x_2 \bmod L$, and $c_1 + s_3x_3 = a_2 + s_1x_3 \bmod L$; two elements with the same slope s_i must be in the same row, which gives $a_1 = a_2$, $b_1 = b_2$, and $c_1 = c_2$. Therefore, this arbitrary cycle-6 is only related to three lines on the lattice: $y = a_1 + s_1x \bmod L$, $y = b_1 + s_2x \bmod L$, $y = c_1 + s_3x \bmod L$, which have common points $(x_1, a_1 + s_1x_1 \bmod L)$, $(x_2, b_1 + s_2x_2 \bmod L)$, and $(x_3, c_1 + s_3x_3 \bmod L)$ and hence form a triangle.

On the other hand, to prove the proposition from the reverse direction, we can pick an arbitrary triangle on the lattice. Assume that there are three lines $y = a + s_1x \bmod L$, $y = b + s_2x \bmod L$ and $y = c + s_3x \bmod L$, which have common points (x_1, y_1), (x_2, y_2) and (x_3, y_3), where $s_1 \neq s_2 \neq s_3$ and $x_1 \neq x_2 \neq x_3$. Since $y_1 = a + s_1x_1 \bmod L = b + s_2x_1 \bmod L$, $y_2 = b + s_2x_2 \bmod L = c + s_3x_2 \bmod L$, and $y_3 = c + s_3x_3 \bmod L = a + s_1x_3 \bmod L$, we have $s_1x_1 + s_2x_2 + s_3x_3 = s_2x_1 + s_3x_2 + s_1x_3 \bmod L$, which satisfies (5.2) in Section 5.2. Therefore, the triangle formed by the three lines can be mapped to a cycle-6 in **H**, which is in the block cycle-6 $P^{-s_1x_1} \to P^{-s_2x_1} \to P^{-s_2x_2} \to P^{-s_3x_2} \to P^{-s_3x_3} \to P^{-s_1x_3}$. □

A triangle is highlighted in Fig. 5.3. Note that Proposition 5.1 is true regardless of the

value of L, which could be prime or not.

5.3.2 Lattice construction with nonprime L

For magnetic recording, we need LDPC codes with appropriate length to encode data sectors. So, it is necessary to extend the lattice construction technique to nonprime L. As mentioned in [66], on the lattice with nonprime L, slopes s should be 0 or co-prime to L such that the lines $y = a + sx \bmod L$ can reach all L values of y with x increasing from 0 to $L-1$. But it is not enough. To prevent the cycle-4's during lattice construction, we need to make sure any two lines starting from the same point $(0, a)$ have no common points except $(0, a)$, if they are of different slopes.

Proposition 5.2: On a rectangular integer lattice $LA = \{(x, y): 0 \le x \le K - 1, 0 \le y \le L - 1\}$, two lines $y = a + s_1 x \bmod L$ and $y = a + s_2 x \bmod L$ with $s_2 > s_1$ have no common points other than $(0, a)$, if $(s_2 - s_1)$ is co-prime to L.

Proof: Let $y_1 = a + s_1 x + r_1 L$ and $y_2 = a + s_2 x + r_2 L$. For a common point, $y_1 = y_2$ gives $(r_1 - r_2) = (s_2 - s_1)\, x / L$. If $(s_2 - s_1)$ is co-prime to L, the only solution for $x \in \{0, 1, \ldots, L - 1\}$, which makes $(r_1 - r_2)$ be an integer, is $x = 0$. □

Remark 5.1: If L is an even number, the slopes co-prime to L are odd numbers. However, the difference of any two odd numbers is an even number, which cannot be co-prime to L. Taking the slope $s = 0$ into account, we conclude that, to construct a lattice LDPC code with column weight equal to or greater than three, L can only be an odd number.

5.3.3 Constructing lattice LDPC codes for magnetic recording

In this chapter, we are interested in the LDPC codes, which have column weight three and rate around 0.9. In other words, we need three slopes on a lattice with $K = 30$. To

encode 4096-bit long sectors, the smallest odd number that L can be is 153. Then, on the rectangular integer lattice with L = 153 and K = 30, we search the three slopes which provide the smallest number of triangles (cycle-6's) by the following procedure.

1) Find the slope set S_1, which include the slope 0 and all slopes co-prime to L. Then the differences of any two slopes in the set S_1 are examined, while a pair of slopes is recorded in the set S_p, whenever the absolute value of their difference is co-prime to L.

2) For each pair of slopes (s_0, s_1) in S_p:

a) Find all slopes s_2, such that the absolute values of $(s_2 - s_0)$ and $(s_2 - s_1)$ are co-prime to L, and record them in the set S_2. Then setup and initialize a counter for each slope in S_2, which will count the number of triangles generated by each three-slope combination (s_0, s_1, s_2).

b) Let l_0 be a line with slope s_0 and l_1 a line with slope s_1. For each pair of lines (l_0, l_1):

i) If l_0 and l_1 do not have a common point, which means that they cannot generate any triangles, then try the next pair of (l_0, l_1).

ii) Otherwise, for any point p_0 on l_0 and any point p_1 on l_1, where $p_0 \neq p_1$, compute the slopes of the lines passing p_0 and p_1, if any. (The method to compute the slopes for any two points on the lattice will be discussed later.) If the slopes for (p_0, p_1) are in the set S_2, then increment the related counters by one.

c) After all pairs of (l_0, l_1) have been examined; the number of triangles for slope combinations (s_0, s_1, s_2) will be transferred from the counters for S_2 to an output buffer.

3) After all pairs of slopes (s_0, s_1) in S_p have been examined; the slope combination,

which produces the smallest number of triangles, will be selected from the output buffer. Finally the **H** matrix in (5.5) will be constructed for the chosen slopes.

Note that this algorithm may compute the triangles for some slope combinations (s_0, s_1, s_2) more than one time; but it is still very efficient.

Now let us discuss the calculation of slopes $0 \le s \le L - 1$ in the Step *2-b*. Let (x_1, y_1) and (x_2, y_2) be two points on the lattice, where $x_2 > x_1$. The slopes of the lines passing through these two points are computed by $s = (y + rL) / x$, where $y = (y_2 - y_1) \bmod L$ and $x = (x_2 - x_1) > 0$.

Proposition 5.3: In the calculation of slopes using $s = (y + rL) / x$, there is exactly one solution for s if x is co-prime to L; there are multiple solutions for s if GCD $(x, L) > 1$ and y is divisible by GCD (x, L); there is no solution for s if GCD $(x, L) > 1$ and y is not divisible by GCD (x, L).

Proof: Let $y = Qx + R$, where $0 \le R < x$. Then $s = (Qx + R + rL) / x = Q + (R + rL) / x$. If x is co-prime to L, $rL \bmod x$ can be any value in $\{0, 1, \ldots, x - 1\}$ with appropriate choices of r. Thus, there are integer solutions for s given appropriate choices of r. On the other hand, given $s_1 = (y + r_1 L) / x$ is an integer, $s_2 = (y + (r_1 + r_2) L) / x = s_1 + r_2 L / x$. If x is co-prime to L, s_2 is an integer if and only if r_2 is a multiple of x, which means $s_1 = s_2 \bmod L$. Thus, there is exactly one solution for s in $\{0, 1, \ldots, L - 1\}$, if x is co-prime to L.

Given a = GCD $(x, L) > 1$, L and x could be expressed as $L = ab$ and $x = ac$. If y is divisible by a, i.e. $y = Qa$, then $s = (Qa + rab) / ac = (Q + rb) / c$. Since b is co-prime to c, s has exactly one solution s_1 in $\{0, 1, \ldots, b - 1\}$, while $s_i = s_1 + r_1 b < L$ with $r_1 > 0$ are also solutions of s.

Similarly, given a = GCD $(x, L) > 1$, L and x could be expressed as $L = ab$ and $x = ac$. If

y is not divisible by a, i.e. $y = Qa + R$, where $0 \leq R < a$, then $s = (Qa + R + rab) / ac = ((Q + rb)\, a + R) / ac$. Obviously for any value of r, $(Q + rb)\, a + R$ cannot be a multiple of a, and hence there is no solution for s. □

On the lattice with $L = 153$, $K = 30$, we find that the smallest number of triangles is 20502 and there are 288 three-slope combinations which give the smallest number of triangles, while {0, 1, 26} is one of such slope combinations. The 459 × 4590 parity-check matrix **H** constructed from slopes {0, 1, 26} gives a (4590, 4133) (3, 30)-regular QC-LDPC code. Although there are 37 bits of overhead, it is the best we can do by lattice construction.

5.3.4 Simulations

To compare the code performance, we need to design a PEG-LDPC code whose parity-check matrix is of the same size as that of the lattice LDPC code. After ten trials of the PEG algorithm, a 459 × 4590 parity-check matrix with column weight three is constructed, which has 24395 cycle-6's and gives a (4590, 4131) PEG-LDPC code.

Shown in Figs. 5.4 and 5.5 are the simulation results for the PEG-LDPC code and the lattice LDPC code at channel densities 0.9713 and 1.3598. Although the lattice LDPC code has fewer cycle-6's, its performance is much worse than the PEG-LDPC code. Actually, there are a lot of undetected errors in the simulation of the lattice LDPC code, which lead to a very high error floor. The reason for the poor performance of the lattice code may have something to do with its deterministic structure, which causes the small minimum distance of the LDPC code, but we are only interested in the number of short cycles in this work. We find that the lattice LDPC code has 2887722 cycle-8's, while the PEG-LDPC code has only 1458723 cycle-8's. We believe that the huge number of cycle-8's in the lattice code plays a

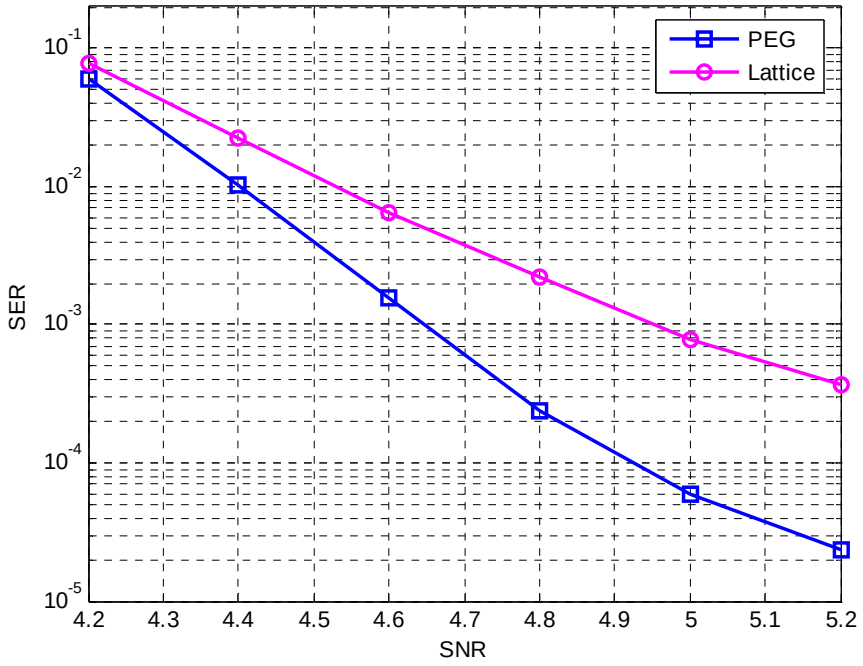


Fig. 5.4. Performance of the lattice LDPC code with $L = 153$ at channel density 0.9713.

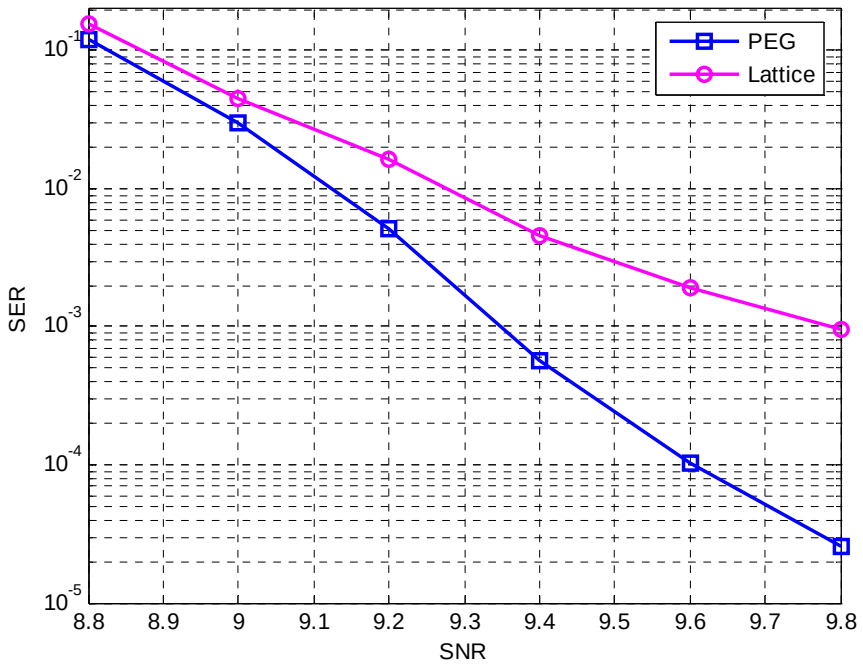


Fig. 5.5. Performance of the lattice LDPC code with $L = 153$ at channel density 1.3598.

role in determining its poor performance. On the other hand, so many cycle-8's could also be an expression of other unknown bad features of the code structure.

5.4 Conclusion

In this chapter, we designed LDPC codes for magnetic recording with many fewer short cycles. The LDPC codes constructed by the MPEG algorithm and Algorithm 5.1 provided remarkable performance improvement over the PEG-LDPC code, and we believe this is due to the fewer short cycles and the randomness of the code construction technique. In addition, we also constructed LDPC codes with fewer cycle-6's based on rectangular integer lattices, using a deterministic construction method. Although we were able to design a lattice LDPC code with fewer cycle-6's than the PEG-LDPC code with a parity-check matrix of the same size, it exhibited a huge number of cycle-8's, which severely degraded its performance. The investigation of the role of code structure features other than short cycles and the connections between other structural features and short cycles is a current area of active research.

6 RS Plus LDPC Codes for Perpendicular Magnetic Recording

It is well known that LDPC codes significantly outperform the traditional RS codes on MRCs. However, there are two difficulties with the replacement of the RS codes. First, we have not only random noise, but also media defects that cause burst errors. While RS codes are guaranteed to correct a fixed maximum number of symbol errors of both types, LDPC codes are usually designed to correct random errors, and they cannot give any assurance on correcting a given number of errors and hence are considered unreliable for media defects. Second, the SER required for MRCs is around 10^{-15}, and LDPC codes may exhibit error floors at high SNRs; their coding gains over RS codes may vanish with increasing SNR before the target SER is achieved. Recently, the use of outer RS codes concatenated with inner LDPC codes has been suggested to get better performance and better reliability [65], [67].

In this chapter, we investigate RS plus LDPC architectures for PMRCs. We consider the concatenation of outer RS codes with different error correction capabilities and inner LDPC codes with various column weights and code rates. To achieve the best decoding performance with a reasonable complexity, we find an optimal iterative decoding scheme, which consists of a maximum number of inner (LDPC decoder) iterations and a maximum number of outer (turbo) iterations. At a fixed user density, the code rate determines the tradeoff between the error correction capability and the channel density penalty. For the optimal iterative scheme, we get the optimal code rates for the concatenated codes by simulation, and evaluate the performance of the concatenated codes in the waterfall region in both random noise and in a media defect scenario. We identify the respective

contributions of the outer RS and inner LDPC codes and observe that the outer RS codes are lowering the error floors of the inner LDPC codes. It is of interest to find the error floors of the concatenated codes, however, the performance estimation of the concatenated code in the error floor region is difficult to compute. At present, we cannot do it for all concatenated codes, but we are able to estimate the error floors for those with inner LDPC codes of column weight two, using the microscopic method proposed in [68].

In Section 6.1, we specify the channel model and the system diagram. In Section 6.2, we design a group of concatenated codes with different combinations of outer RS codes and inner LDPC codes. We find the optimal iterative scheme and code rates in Section 6.3, and compare and discuss the waterfall region performance of the coded channels in Section 6.4. In Section 6.5, we estimate the error floors of the concatenated codes which have inner LDPC codes with column weight of two. Finally, we conclude the paper with a summary of the results and suggestions for additional work in Section 6.6.

6.1 Channel model

In this chapter, we still use the PMRC model and the corresponding SNR definition specified in Chapter 1, while we consider a mix of 50% jitter noise power and 50% electronics noise power in the read channel. Note that we are purposely using a channel with a low percentage of jitter noise power to mimic a channel with a high percentage of jitter noise power but with a data-dependent noise predictive detector [69]. In this chapter, we only consider channels with a moderate user density of 1.049.

We show in Fig. 6.1 the system diagram of the coded PMRC, where SOVA is utilized for channel detection, and the normalized MS algorithm [45]-[47], [70], implements the

LDPC decoder. The outer RS code is decoded by hard decision decoding.

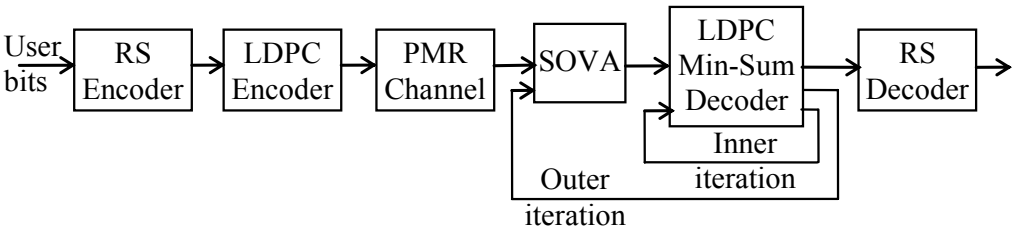


Fig. 6.1. System diagram for an RS plus LDPC coded PMRC.

Table 6.1 RS Plus LDPC Codes

t of outer RS code	Inner LDPC code rate (≈)	Inner LDPC (n, k)	Overall code rate
16	0.84	(5262, 4420)	0.7792
	0.86	(5140, 4420)	0.7977
	0.88	(5023, 4420)	0.8162
	0.90	(4911, 4420)	0.8349
20	0.84	(5357, 4500)	0.7654
	0.86	(5233, 4500)	0.7835
	0.88	(5114, 4500)	0.8017
	0.90	(5000, 4500)	0.8200
24	0.84	(5452, 4580)	0.7520
	0.86	(5326, 4580)	0.7698
	0.88	(5205, 4580)	0.7877
	0.90	(5089, 4580)	0.8057

6.2 Concatenated code design

We design (442, 410), (450, 410) and (458, 410) shortened RS codes over GF (2^{10}) as outer codes, which we denote by RS (t = 16), RS (t = 20) and RS (t = 24) respectively, where t is the error correction capability. For each outer RS code, inner LDPC codes with

different rates (R = 0.84, 0.86, 0.88 and 0.9) are designed by the PEG algorithm [50]. For each inner code rate, we construct two LDPC codes, with parity-check matrices of constant column weight two (W_c = 2) and three (W_c = 3), respectively. We list the concatenated codes designed in Table 6.1, where the information word lengths of the inner LDPC codes match the lengths of the outer RS codes.

6.3 Optimal iterative scheme and code rate

We use turbo equalization to improve decoding performance, and simulate the concatenated codes at appropriate SNRs under various iterative schemes, where each iterative scheme consists of a particular combination of the maximum number of inner and outer iterations. In Fig. 6.2 we show the iterative scheme test for the RS (t = 16) + LDPC (R=0.88, W_c=2) code at SNR=8 dB. We observe that additional outer iterations always improve performance, while more than six inner iterations only give a marginal improvement. All other concatenated codes in this work, for different t, R, and W_c, have similar graphs as in Fig. 6.2. Therefore, given that the decoding complexity of the system needs to be kept at a reasonable level, we chose the iterative scheme with at most six outer (turbo) iterations and six inner iterations, denoted by T6MS6, as the one for all concatenated codes.

At a fixed user density, lowering the code rate may enhance the error correction capability of the codes, but increases the channel density penalty. To find the optimal tradeoff, we vary the inner code rates for each outer RS code and measure their decoding

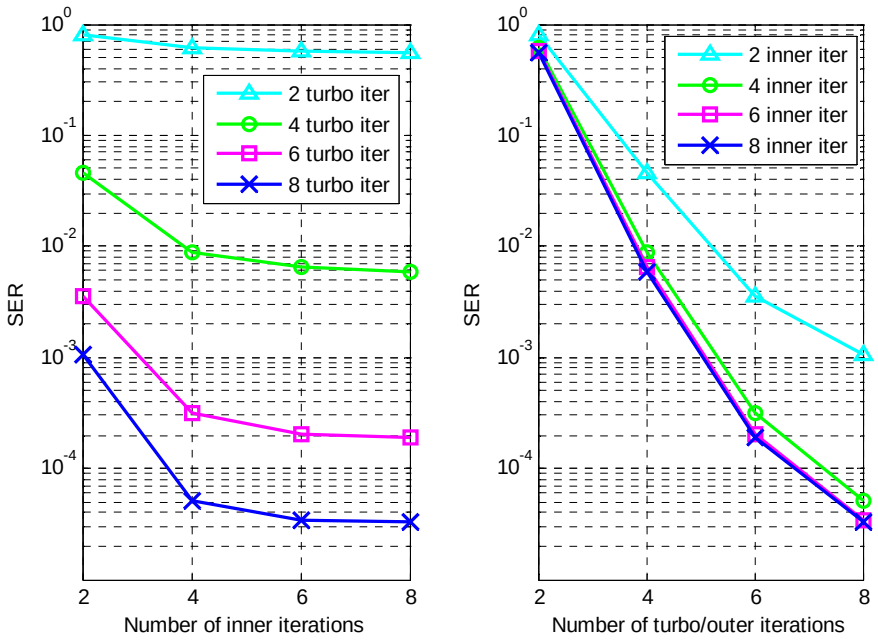


Fig. 6.2. Performance of RS (t = 16) + LDPC (R=0.88, W_c=2) code at SNR=8 dB, under different iterative schemes.

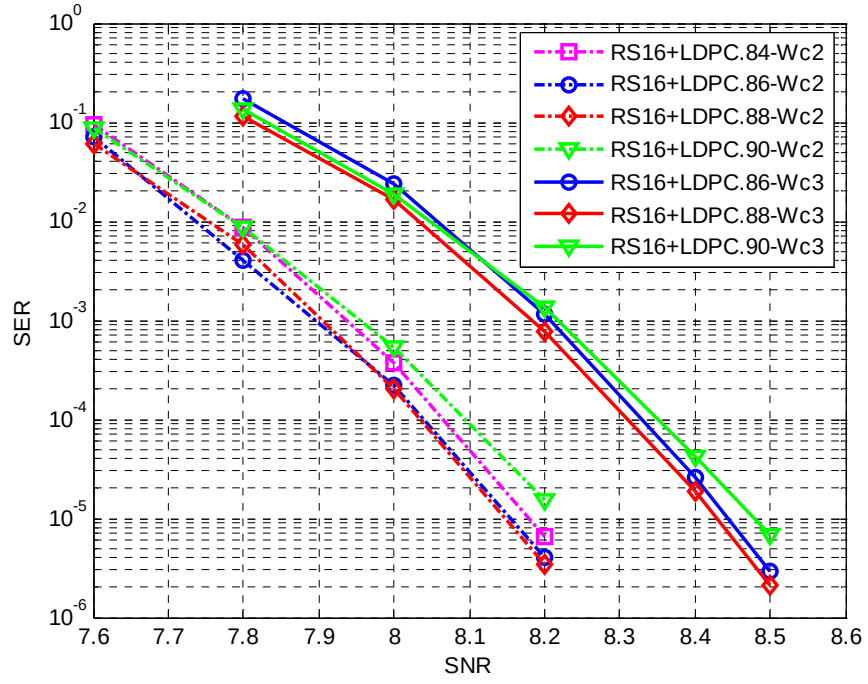


Fig. 6.3. Performance of RS (t = 16) + LDPC codes with different code rates and column weights.

performance under the selected iterative scheme (T6MS6). We show in Fig. 6.3 the performance curves for RS (t = 16) + LDPC codes with different rates and column weights for the inner LDPC codes. We can see that, for each column weight, the inner LDPC code with a rate around 0.88 exhibits the best performance, and further increasing the inner code rate severely degrades their error correction capability. For the concatenated codes with RS (t = 20) and RS (t = 24), this is also the case, although we are not presenting those simulation results. In other words, the optimal code rates of the concatenated codes solely depend on the inner code rate: given a pair of t and W_c, the concatenated codes with inner code rate of 0.88 always provide the best performance.

6.4 Performance of RS plus LDPC codes

With the optimal iterative scheme (T6MS6), we simulate concatenated codes with inner code rate of 0.88 on PMRCs with both random noise and media defects. The media defects here consist of 50 bits of half erasure and their locations are assumed to be known and available to the decoder. For comparison purposes, we design two (4655, 4096) LDPC codes of rate 0.88, which have constant column weights of two and three, respectively. We simulate the two LDPC-only coded PMRCs with the same iterative scheme (T6MS6) and use them as base lines.

We can see in Figs. 6.4 and 6.5 that the concatenated codes provide remarkable gains over the RS-only (t = 24) code in both noise environments, and that the LDPC code with lower column weight exhibits better performance. Due to the severe channel density penalty, stronger outer RS codes lead to worse performance in the simulation range considered. However, the advantage of the concatenated codes is also obvious. On the one

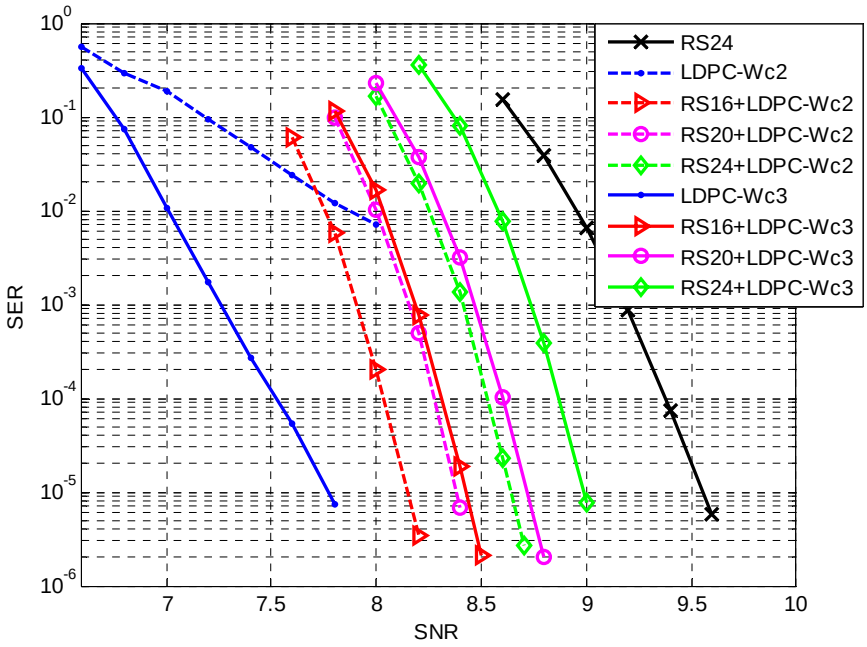


Fig. 6.4. Performance of RS + LDPC (R=0.88) codes in random noise for different RS and LDPC codes.

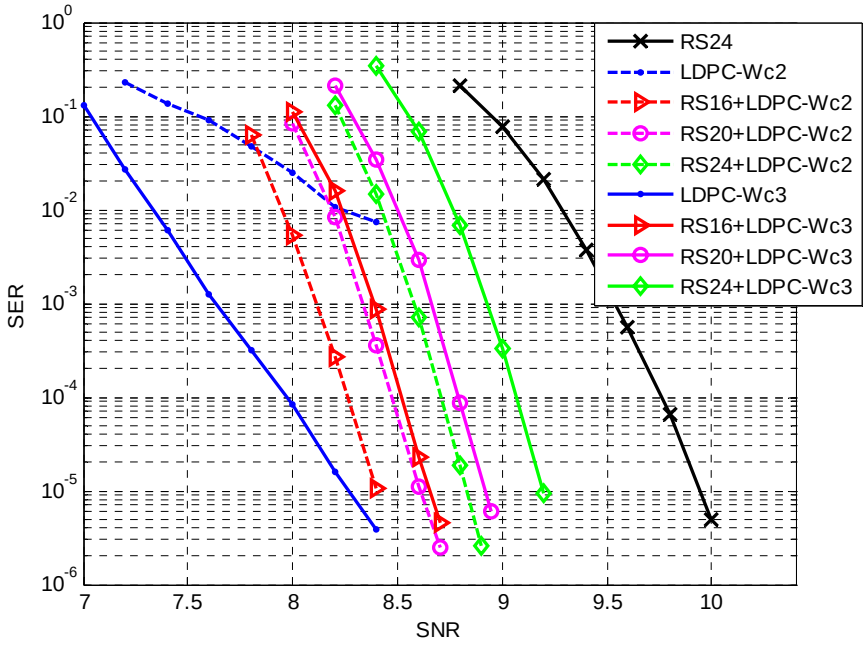


Fig. 6.5. Performance of RS + LDPC (R=0.88) codes with media defects for different RS and LDPC codes.

hand the LDPC-only code with column weight of two exhibits a high error floor, and so do the inner LDPC codes with column weight of two. Since the concatenated codes always have sharp curves in our simulations, it is clear that the outer RS codes are lowering the error floors caused by the inner LDPC codes. Similarly, the concatenated codes have shaper curves than the LDPC-only codes with column weight of three, and the concatenated codes are expected to have better performance than LDPC-only codes at high SNR. On the other hand, the concatenated codes achieve larger coding gain over the RS-only (t = 24) code in the presence of media defects than in random noise, while the coding gains of the LDPC-only code over the RS-only (t = 24) code become smaller in the presence of media defects. In other words, the concatenated codes are the most robust codes and exhibit a smaller performance degradation in the presence of media defects than the LDPC-only and RS-only codes.

6.5 SER estimation of RS plus LDPC codes

From Figs. 6.4 and 6.5, we note that the best performance is obtained for the LDPC-only code with column weight three. But as mentioned in the introduction, LDPC codes may have error floors at high SNR. To do a complete evaluation, the performance of the LDPC-only code and concatenated codes should be compared at higher SNRs, where very low SERs are expected. Unfortunately, at present, we cannot estimate the performance of the LDPC-only (W_c=3) code at high SNRs, especially under this complicated iterative scheme. For the concatenated codes, the multinomial model is widely used to estimate the SER at the output of RS decoder [71], [72], but we find that the multinomial model cannot give an accurate estimation for the RS plus LDPC codes. Recently, Kuznetsov *et al.*

proposed a microscopic approach [68] to estimate the SER of RS-plus codes more accurately, based on the distributions of both error event weight and the number of error events in a sector.

6.5.1 Microscopic method

In this subsection, we briefly review the basic concept of the microscopic method. Let p_l denote the probability of an error event that contains l RS symbol errors. We can estimate p_l by simulation, which we denote as $\hat{p}_l$, with $l = 1, \ldots, L$. Then we use a Markov chain of order M to model the probability mass function (PMF) $\hat{p}_l$ and then generate p_l, and optimize the model parameters by minimizing the Kullback–Leibler distance between the PMF $\hat{p}_l$ and the weight distribution produced by the model. Using the Markov chain model, it is easy to compute the conditional word failure rate $\rho_W(t, k)$, which is the probability of the occurrence of more than t RS symbol errors in a sector, given that there are k error events observed.

We then model the distribution of the number of error events in a sector by tail fitting. At first, one can get a PMF $Q(k)$ of the number of error event k by simulation, where $Q(k) = 0$ for k greater than some value J, due to limited simulation time. For an LDPC code, we use an exponential function $\hat{Q}(k) = \alpha e^{-\lambda k}$ with $k > J$ to re-construct the missing tail of $Q(k)$. The tail-fitting finds the values of α and λ which minimize the quantity

$$\sum_{k=k_0}^{J} \left| \left(\hat{Q}(k) - Q(k) \right) \Big/ \hat{Q}(k) \right|^2 , \tag{6.1}$$

where $0 < k_0 \le J$. Then $\tilde{Q}(k)$, the distribution of the number of error events, is made by combining $Q(k)$ and $\hat{Q}(k)$ together, where $\tilde{Q}(k) = Q(k)$ for $k \le J$ and $\tilde{Q}(k) = \hat{Q}(k)$ for k

> J. Finally, the SER of RS plus LDPC codes is computed by

$$\rho_W(t) = \sum_{k=0}^{\infty} \tilde{Q}(k)\rho_W(t,k). \tag{6.2}$$

6.5.2 SER estimation

To accurately estimate the SER of RS plus LDPC codes by the microscopic method, it is necessary to obtain enough number of error events by simulation to find the distributions $\hat{p}_l$ and $Q(k)$. Unfortunately, we cannot get enough error events for RS + LDPC (W_c=3) codes; at this time, we are only able to estimate the SER for RS + LDPC (W_c=2) codes by the microscopic method. We have shown in Section 6.4 that RS + LDPC (W_c=2) codes provide remarkable coding gains over the RS-only code. It is interesting to verify if these gains vanish at high SNR.

By simulating the RS + LDPC (W_c=2) codes in a wide SNR region, we obtain enough number of error events to draw PMFs $\hat{p}_l$ and $Q(k)$. However, we find that $Q(k)$ may not have a simple exponential tail. For example, we show in Fig. 6.6 the distributions $Q(k)$ for the RS (t = 16) + LDPC (W_c=2) code at different SNRs. At 7.6 dB, $Q(k)$ has two corners which separate the distribution curve into a sharp left region, a flat middle region and a sharp tail region, where the fluctuation in the sharp tail region is caused by the big estimation variation at large k. At higher SNRs, we lose track of the sharp tail region and we are also gradually missing the flat middle region. At very high SNRs (higher than 9 dB), we are only able to detect the head region of $Q(k)$, which exhibits a zigzag pattern just as mentioned in [68].

Since $Q(k)$ does not have a simple exponential tail, the tail-fitting using (6.1) may not lead to an accurate estimation of the SER. However, we note that $\rho_W(t, k)$ is a

non-decreasing sequence with k going to infinity. If we do the exponential tail fitting only for the head region and ignore the flat middle region of $Q(k)$ at high SNR, we get an estimator, which gives a lower bound on the decoding performance. Since the existence of the flat middle region is assumed but cannot be proved in this work, the lower bound generated by this estimator is only a reasonable conjecture.

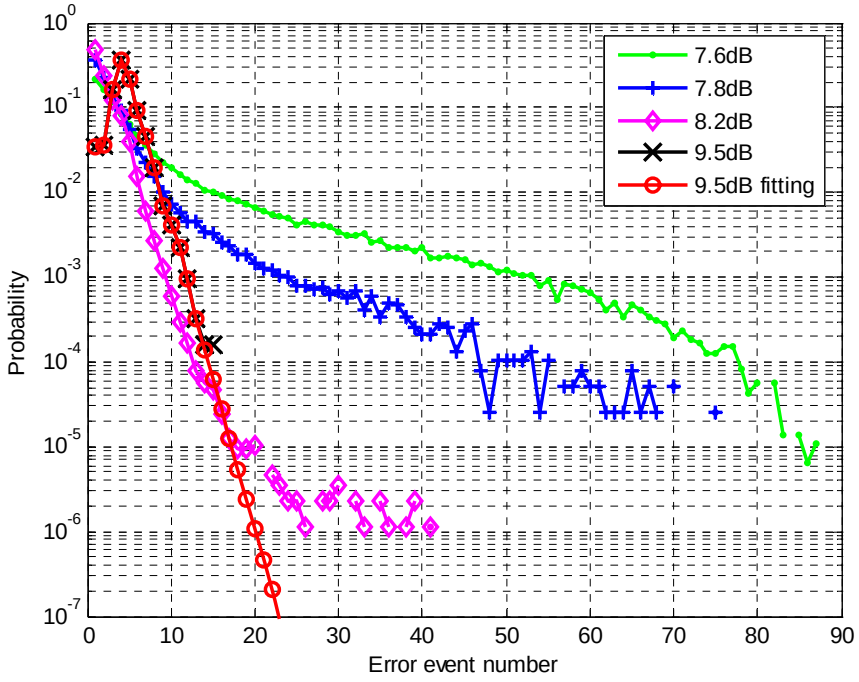


Fig. 6.6. PMFs $Q(k)$ for RS (t = 16) + LDPC (R=0.88,W_c=2) code at different SNRs and $\tilde{Q}(k)$ by exponential tail-fitting at SNR=9.5 dB.

We show in Fig. 6.7 the SER estimates for RS (t = 16, 20, 24) + LDPC (W_c=2) codes. The estimator provides accurate results at low SNRs and is expected to give a lower bound at high SNRs. We expect these concatenated codes to have flat error floors, because their inner LDPC (W_c=2) codes also have flat error floors. In addition, the concatenated codes with larger t exhibit a lower error floor, which explains the contribution of the outer RS

codes. We also observe that the performance lower bounds are above the SER of 10^{-15}; if the target working SER of the system is 10^{-15}, then RS (t = 16, 20, 24) + LDPC (W_c=2) codes may not be good choices.

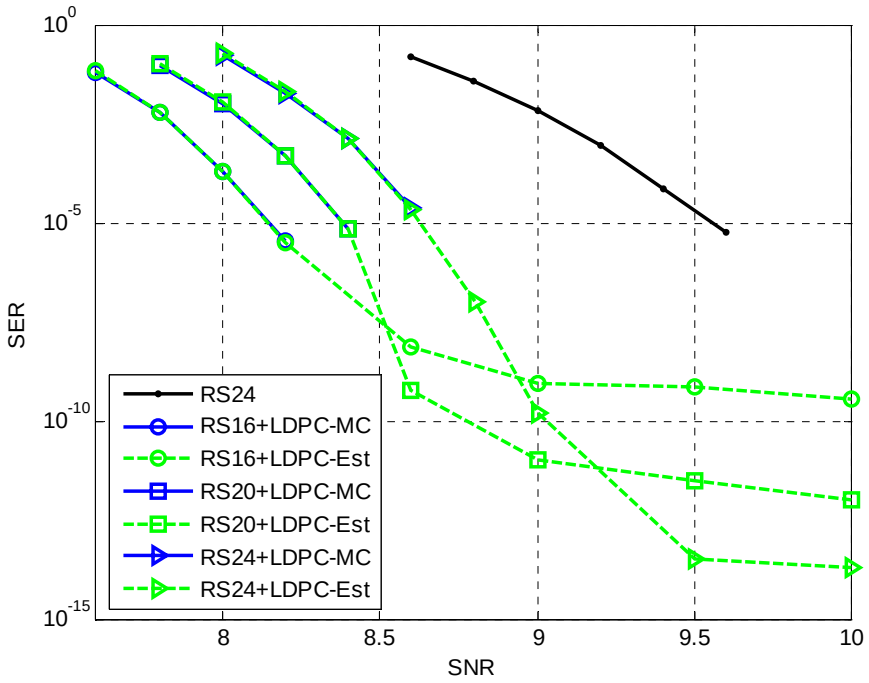


Fig. 6.7. SER estimation for RS (t = 16, 20, 24) + LDPC (R=0.88, W_c=2) codes.

6.6 Conclusion

In this chapter, we investigate RS plus LDPC architectures for PMRCs in great detail, and find the best iterative scheme and code rate. We compare the waterfall region performance for various RS plus LDPC codes in both random noise and with media defects, and estimate the error floor for the RS + LDPC (W_c=2) codes. The interesting question of estimating the performance of both LDPC-only and the RS + LDPC (W_c=3) codes at high SNR and compare them with our simulation results remains open.

7 Multi-Track Detection for Inter-Track Interference Mitigation

To keep up with the increasing demand for data storage, the recording density of future storage systems needs to continue to increase. Continuous media PMR has physical and engineering limitations, which prevents its use at extremely high recording densities. BPMR is a promising technology which is expected to enable the high density recording up to four Terabits per square inch (Tb/in^2). However, while BPMR offers some advantages over the conventional PMR currently in use, it also presents new challenges from the read-channel design perspective and specific impairments, such as write errors, side readings, island size, location, and shape variations.

High recording density is achieved by reducing the inter-track distance, which causes side reading and the associated ITI becomes a major BPMR-specific impairment. Attempts to mitigate the impact of ITI on the sector error rate performance, include the joint-track equalization proposed in [73], which optimizes a one-dimensional (1D) equalizer with a 2D GPR target, which was shown to provide some gain over the conventional single-track equalization (with 1D GPR targets), in the presence of strong ITI. Although the investigation of the joint-track equalization in [73] was done for continuous media, we believe it to be applicable to BPMR channels as well. Another approach considered in [74] is 2D equalization, where a 2D equalizer and a 1D GPR target are simultaneously optimized. Although the 2D equalization has been shown to achieve significant gains over single-track equalization, we note that none of these two equalization methods takes full advantage of the channel detection with 2D GPR targets, which we consider a very useful technique to mitigate the impact of ITI.

In this chapter, we revisit the equalizer and 2D GPR target design for both joint-track and 2D equalizations. By investigating the detection with 2D GPR targets from a new point of view, we find that during detection on a desired track, it is necessary to estimate the data on the side tracks to fully benefit from the equalization with 2D GPR targets, and this can be accomplished by using multi-track detection.

The rest of this chapter is organized as follows. In Section 7.1, we describe the BPMR channel model used. Before investigating any advanced equalization techniques, we introduce the single-track equalized BPMR channel in Section 7.2. In Section 7.3, we investigate the joint-track equalized BPMR channel as well as the detection on the trellis of a 2D GPR target. Then we explain the relationship between detection performance and mean-squared equalization error, from a new perspective. In Section 7.4, we propose a multi-track detection technique and apply it to the joint-track equalized channel. In Section 7.5, we use the proposed technique on a 2D equalized BPMR channel. In Section 7.6, we obtain the performance bounds for multi-track detection for channels with joint-track and 2D equalizations. Then we propose an extension of the multi-track detection technique to achieve the performance bounds. In Section 7.7, we perform simulations on media with two different island distributions to further validate the proposed techniques. We conclude this chapter with a brief discussion of the main results in Section 7.8.

7.1 Channel model

The BPMR channel is characterized by the replay response to an isolated magnetic island, which is affected by different media types and geometric characteristics of the island and the read head. In this chapter, we only consider perpendicular patterned media

with soft underlayers. In [13], [75] and [76], Nutter *et al.* simulated the 2D replay response of a single island by a 3D reciprocity integral, which takes into account the 3D geometry of the island. Using this simulation method on a perpendicular patterned medium with a soft underlayer, as shown in [76], the replay response of a square island with length a = 12.5 nm has a pulse width (at half maximum) PW_{50_along} = 21.2 nm on the along track direction and a PW_{50_cross} = 31.2 nm on the cross track direction, where the film thickness is δ = 10 nm and the giant-magnetoresistive (GMR) read head has dimensions: sensor width W = 20 nm, sensor length L = 4 nm, shield-to-sensor spacing G = 6 nm, and a fly height d = 10 nm. Nabavi *et al.* use a similar but different way to simulate the 2D response of a perpendicularly magnetized island [14], [20]. As in [20], the replay response of a square island with length a = 11 nm has PW_{50_along} = 19.5 nm and PW_{50_cross} = 24.7 nm, where the employed medium and read head dimensions are δ = 10 nm, W = 15 nm, L = 4 nm, G = 6 nm, and d = 10 nm.

During the modeling of the read channel, we need a high resolution replay response. The simulation methods proposed by Nutter *et al.* and Nabavi *et al.* have high computational complexity, especially when media noises are considered [14], [20]. For modeling simplicity, a 2D replay response can be approximated by a close form function as $h(x, y) = h_x(x)h_y(y)$, where $h_x(x)$ and $h_y(y)$ are the analytic functions assumed on the along-track direction and the cross-track direction, respectively. In [77], Keskinoz uses Nutter *et al.*'s read head and medium (with square island a = 12.5 nm) as we mentioned above; but the cross-track profile $h_y(y)$ is simplified to a Lorentzian pulse. Similarly, Nabavi notes that the replay response of the square island a = 11 nm with the read head and medium configuration in [20] can be well fitted by Gaussian pulses on both along-track and

cross-track directions.

Throughout this paper, we employ Nabavi's medium and read head in [20] and the replay response of an isolated square island with a = 11 nm is simplified to a 2D Gaussian pulse,

$$h(x,y) = h_x(x)h_y(y) = A\exp\left(-\frac{1}{2}\left(\frac{c^2x^2}{19.5^2}+\frac{c^2y^2}{24.7^2}\right)\right), \tag{7.1}$$

where A=1 is the peak amplitude; $c = 2\sqrt{2\ln 2}$ is a constant used to associate the PW_{50} to the standard deviation of the Gaussian function. We arrange the islands on rectangular grids and vary the island periods T_x on the along-track direction and T_y on the cross-track direction to achieve different areal densities. To facilitate the explanation and illustration of the equalization and detection techniques investigated in this paper, we carefully choose the track pitch T_y so that ITI is mostly caused only by the two nearest side tracks.

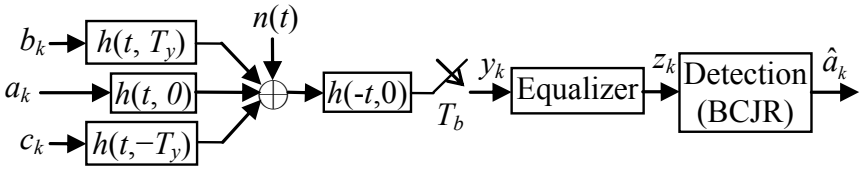


Fig. 7.1. BPMR channel model for single track reading and detection.

Therefore, we obtain a very simple model for the BPMR channel, as shown in Fig. 7.1, where a_k are the data on the desired (center) track; b_k and c_k are the data on the two side tracks; a_k, b_k and c_k are +1 or −1. There is no media noise included in this model. The electronic noise $n(t)$ is additive white Gaussian with double sided power density height of σ^2, and the SNR is defined as $1/\sigma^2$. The low pass filter $h(-t, 0)$ matches the response on the center track $h(t, 0) = h_x(t)$. Since the 2D Gaussian pulse $h(x, y)$ in (1) is symmetric on both

x and y directions, we have $h(-t, 0) = h(t, 0) = h_x(t)$ and $h(t, T_y) = h(t, -T_y) = h_y(T_y)h_x(t) = h_y(T_y)h(t, 0)$. So the response of the side tracks is simply a scaled version of the center track response. To measure the level of ITI in the read channel, we define $A_s = h_y(T_y)$ as the side track amplitude. In addition, to clearly show the extent of ISI on the along-track direction, we define the normalized pulse width $\mathrm{PW}_N = \mathrm{PW}_{50\_\mathrm{along}} / T_x$, so that larger PW_N means stronger ISI in the read channel.

Finally, please note that the diagram given in Fig. 7.1 is only for single-track reading and detection, which is applicable to single-track equalization and to joint-track equalization. However, 2D equalization requires multi-track reading; we will build up a more complicated channel model for 2D equalization, using the one in Fig. 7.1 as the very basic component.

7.2 Single-track equalization

The single-track equalization with optimized GPR targets was initially proposed by Moon *et al.* in [8] and later applied to PMR channels in [9], which is the GPR target equalization method we have introduced on PMRCs in Chapter 1. Recently, Nutter *et al.* [76] use this technique to equalize BPMR channels. It has been proved that the channels equalized with optimized GPR targets, which have non-integer coefficients, significantly outperform the channels equalized with the conventional integer PR targets. In this chapter, we also consider the single-track equalized BPMR channel and use it as the base line in the performance comparison of different equalization-detection techniques.

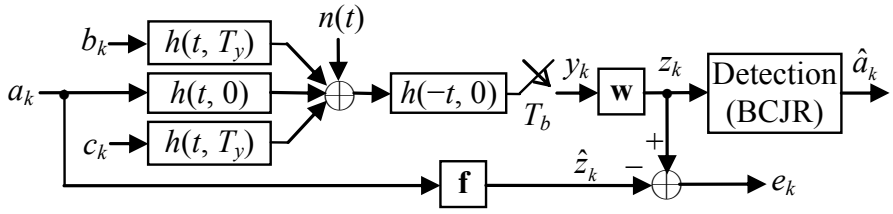


Fig. 7.2. Single-track equalized BPMR channel with optimized GPR targets.

The diagram of the single-track equalized BPMR channel is drawn in Fig. 7.2, where the equalizer $\mathbf{w} = [w_{-N}, \ldots, w_0, \ldots, w_N]^{\mathrm{T}}$ is a $2N$+1 taps FIR digital filter, and $\mathbf{f} = [f_0, \ldots, f_{L1-1}]^{\mathrm{T}}$ is the GPR target with length L_1. Although the GPR target $\mathbf{f}$ could be anti-causal, we always use causal targets in this work. Let $\mathbf{R_y}$ be the $(2N+1)\times(2N+1)$ autocorrelation matrix of y_k, with $\mathbf{R_y}(i, j) = E\{y_{k-i}\, y_{k-j}\}$ for $-N \le i, j \le N$, $\mathbf{R_a}$ the $L_1 \times L_1$ autocorrelation matrix of a_k, with $\mathbf{R_a}(i, j) = E\{a_{k-i}\, a_{k-j}\}$ for $0 \le i, j \le L_1 - 1$, $\mathbf{R_{y,a}}$ the $(2N+1)\times L_1$ cross correlation matrix with $\mathbf{R_{y,a}}(i, j) = E\{y_{k-i}\, a_{k-j}\}$ for $-N \le i \le N$ and $0 \le j \le L_1 - 1$. Then the mean-squared error (MSE) can be expressed as

$$\mathrm{MSE_{SE}} = E\left\{e_k^2\right\} = \mathbf{f}^{\mathrm{T}}\mathbf{R_a}\mathbf{f} + \mathbf{w}^{\mathrm{T}}\mathbf{R_y}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{R_{y,a}}\mathbf{f}\,. \tag{7.2}$$

By minimizing (7.2) and enforcing $f_0 = 1$, the optimized GPR target and equalizer can be computed as

$$\mathbf{f} = \lambda(\mathbf{R_a} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,a}})^{-1}\mathbf{C}\,, \tag{7.3}$$

$$\mathbf{w} = \mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,a}}\mathbf{f}\,, \tag{7.4}$$

where $\mathbf{C} = [1, 0 \ldots 0]^{\mathrm{T}}$ is a vector of length L_1, and

$$\lambda = \frac{1}{\mathbf{C}^{\mathrm{T}}(\mathbf{R_a} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,a}})^{-1}\mathbf{C}}\,.$$

We take N =10 to setup an equalizer with 21 taps. Actually we use 21 taps filters as

equalizers not only in the single-track equalization but also for the joint-track equalization and each dimension of the 2D equalization, because 21 taps are large enough to avoid affecting the design of the equalizer and the GPR target, and in addition it is a fair comparison of different recording systems.

7.3 Joint-track equalization

Joint-track equalization [73] is a single-track reading and detection technique, where the equalizer is still 1D, just like the single-track equalization. However, this technique equalizes the channel with an optimized 2D GPR target, i.e., GPR targets for center and side tracks are designed, which achieve a smaller MSE than that of single-track equalization, but need a complicated channel detector. After giving a brief review of the joint-track equalization technique, we will put forward a new interpretation of this well known approach.

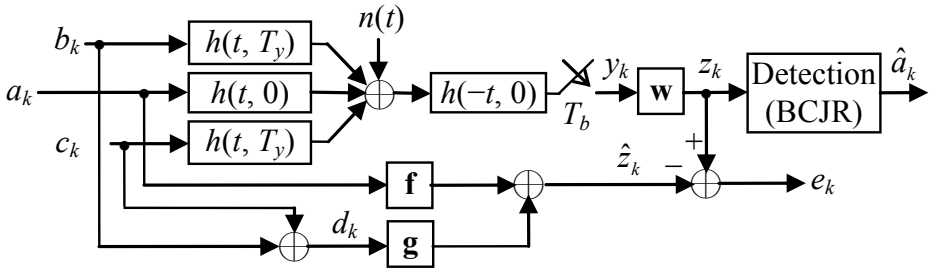


Fig. 7.3. Joint-track equalized BPMR channel with optimized 2D GPR target and 1D equalizer.

7.3.1 Joint-track equalized BPMR channel

The diagram of the joint-track equalized BPMR channel is shown in Fig. 7.3, where we express the data on the two side tracks, b_k and c_k as $d_k = b_k + c_k$, given the equivalent

contributions of b_k and c_k in the channel. To facilitate the derivation of the joint-track equalization, we need to define the target and some statistical quantities related to the input d_k, besides extending the notation for single-track equalization. Let $\mathbf{g} = [g_0, \ldots, g_{L2-1}]$ be the GPR target for the side tracks, $\mathbf{R_d}$ the $L_2 \times L_2$ autocorrelation matrix of d_k, with $\mathbf{R_d}(i, j) = E\{d_{k-i}\, d_{k-j}\}$ for $0 \le i, j \le L_2-1$, $\mathbf{R_{a,d}}$ the $L_1 \times L_2$ cross correlation matrix, with $\mathbf{R_{a,d}}(i, j) = E\{a_{k-i}\, d_{k-j}\}$ for $0 \le i \le L_1-1$ and $0 \le j \le L_2-1$, $\mathbf{R_{y,d}}$ the $(2N+1) \times L_2$ cross correlation matrix with $\mathbf{R_{y,d}}(i,j) = E\{y_{k-i}\, d_{k-j}\}$ for $-N \le i \le N$ and $0 \le j \le L_2-1$. Then the MSE can be expressed as

$$\begin{aligned} \mathrm{MSE}_{\mathrm{JE}} &= E\left\{e_k^2\right\} \\ &= \mathbf{f}^{\mathrm{T}}\mathbf{R_a}\mathbf{f} + \mathbf{g}^{\mathrm{T}}\mathbf{R_d}\mathbf{g} + 2\mathbf{f}^{\mathrm{T}}\mathbf{R_{a,d}}\mathbf{g} + \mathbf{w}^{\mathrm{T}}\mathbf{R_y}\mathbf{w} \\ &\quad - 2\mathbf{w}^{\mathrm{T}}(\mathbf{R_{y,a}}\mathbf{f} + \mathbf{R_{y,d}}\mathbf{g}). \end{aligned} \tag{7.5}$$

By minimizing (7.5) and enforcing $f_0 = 1$, the optimized 2D GPR target and 1D equalizer can be computed as

$$\mathbf{f} = \lambda(\mathbf{A}_1 - \mathbf{B}_1\mathbf{A}_2^{-1}\mathbf{B}_2)^{-1}\mathbf{C} \tag{7.6}$$

$$\mathbf{g} = -\mathbf{A}_2^{-1}\mathbf{B}_2\mathbf{f} \tag{7.7}$$

$$\mathbf{w} = \mathbf{R}_{\mathbf{y}}^{-1}(\mathbf{R_{y,a}}\mathbf{f} + \mathbf{R_{y,d}}\mathbf{g}) \tag{7.8}$$

where $\mathbf{C} = [1,0 \ldots 0]^{\mathrm{T}}$ is a vector of length L_1, as in the single-track equalization, and

$$\lambda = \frac{1}{\mathbf{C}^{\mathrm{T}}(\mathbf{A}_1 - \mathbf{B}_1\mathbf{A}_2^{-1}\mathbf{B}_2)^{-1}\mathbf{C}},$$

$$\mathbf{A}_1 = \mathbf{R_a} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,a}}, \qquad \mathbf{B}_1 = \mathbf{R_{a,d}} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,d}},$$

$$\mathbf{A}_2 = \mathbf{R_d} - \mathbf{R}_{\mathbf{y,d}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,d}}, \qquad \mathbf{B}_2 = \mathbf{R}_{\mathbf{a,d}}^{\mathrm{T}} - \mathbf{R}_{\mathbf{y,d}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R_{y,a}}.$$

This design of the joint-track equalizer has a different expression from that in [73], but they are essentially equivalent, except that we emphasize that the target lengths L_1 and L_2

are not necessarily the same.

7.3.2 Detection on the trellis of a 2D GPR target

Based on the 2D GPR target, a trellis with multiple inputs needs to be constructed for joint-track detection. But we have choices here: the trellis could have either three binary inputs a_k, b_k and c_k, or one binary input a_k and one ternary input d_k. Since b_k and c_k have a statistically identical contribution in the channel and hence have the same optimized GPR target, the trellises with these two kind of inputs, $\{a_k, b_k, c_k\}$ and $\{a_k, d_k\}$ are equivalent with respect to the detection of a_k. With an input of $\{a_k, b_k, c_k\}$, each trellis state needs L_1-1 bits of memory to store $\{a_{k-1}, \ldots, a_{k-L1+1}\}$ and another $2(L_2-1)$ bits memory for $\{b_{k-1}, \ldots, b_{k-L2+1}\}$ and $\{c_{k-1}, \ldots, c_{k-L2+1}\}$. Therefore such a trellis has $2^{(L1-1)}\times 4^{(L2-1)}$ states, and each state has eight outgoing branches corresponding to the eight possible combined inputs of $\{a_k, b_k, c_k\}$. On the other hand, the trellis with input $\{a_k, d_k\}$ needs only $2^{(L1-1)}\times 3^{(L2-1)}$ states, and each state has six outgoing branches corresponding to the six possible combined inputs of $\{a_k, d_k\}$. Therefore, we consider the simpler trellis with the input $\{a_k, d_k\}$ in the following investigation.

A couple of detection methods on a trellis of a 2D target were discussed in [10]. In this chapter, we only consider the optimal detector based on the BCJR algorithm [28], which is named joint-BCJR in [10]. On the trellis with the input $\{a_k, d_k\}$, the forward and backward recursions of the joint-BCJR have the same expressions as those of the standard BCJR algorithm,

$$\alpha_k(s_k) = \sum_{s_{k-1}} \alpha_{k-1}(s_{k-1}) \cdot \gamma(s_{k-1}, s_k), \tag{7.9}$$

$$\beta_{k-1}(s_{k-1}) = \sum_{s_k} \beta_k(s_k) \cdot \gamma(s_{k-1}, s_k), \tag{7.10}$$

where $\alpha_k(s_k)$ and $\beta_k(s_k)$ are the forward and backward state probabilities, respectively. The branch transition probability $\gamma(s_{k-1}, s_k)$ is computed by

$$\gamma(s_{k-1}, s_k) = P(a_k, d_k) \cdot P(z_k \mid s_{k-1}, a_k, d_k, s_k), \tag{7.11}$$

where the *a priori* probability $P(a_k, d_k) = P(a_k)P(d_k)$, and $P(d_k)$ is related to $P(b_k)$ and $P(c_k)$ by $P(d_k=-2) = P(b_k=-1)P(c_k=-1)$, $P(d_k=2) = P(b_k=1)P(c_k=1)$ and $P(d_k=0) = P(b_k=-1)P(c_k=1) + P(b_k=1)P(c_k=-1)$. Then the APP is computed as

$$P(a_k, d_k \mid \mathbf{z}) = \frac{1}{P(\mathbf{z})} \sum_{s_{k-1}} \alpha_{k-1}(s_{k-1}) \cdot \gamma(s_{k-1}, s_k) \cdot \beta_k(s_k), \tag{7.12}$$

where $\mathbf{z}$ is a block of the received signal. Finally, the desired APP for a_k is obtained by marginalization,

$$P(a_k \mid \mathbf{z}) = \sum_{d_k} P(a_k, d_k \mid \mathbf{z}). \tag{7.13}$$

7.3.3 Performance and mean-squared error

As shown in Figs. 7.2 and 7.3, the single-track equalization simply treats the ITI as an additive noise, while the joint-track equalization designs an optimized PR target for side tracks, given that the input distribution of side tracks is known, which is expected to be very useful information to the receiver. Now we would like to do some simulations to see if the joint-track equalization can achieve significant gains over the single-track equalization on BPMR channels.

In this section, we choose an arrangement of islands with T_x = 13 nm, T_y = 18.82 nm, which gives an areal density of 2.64Tb/in^2; such a bit-patterned medium is characterized by strong ISI and ITI, where the normalized pulse width is PW_N = 1.5 and the side track amplitude is A_s = 0.2. We simulate the uncoded read channels of both equalization methods

with 4096-bit sectors, where the single-track equalized channel has GPR3 targets and the joint-track equalized channel has a GPR3 target for the center track and GPR2 targets for the side tracks. We use shorter targets on the side tracks to reduce the detection complexity, while we find that targets longer than two taps for side tracks give only negligible improvement on the minimum MSE (MMSE) and the bit-error-rate (BER) performance. We present the simulation results in Fig. 7.4, and it is a bit surprising that the two equalization methods provide similar BER performance, since we observe that joint-track equalization has a much smaller MSE than single-track equalization.

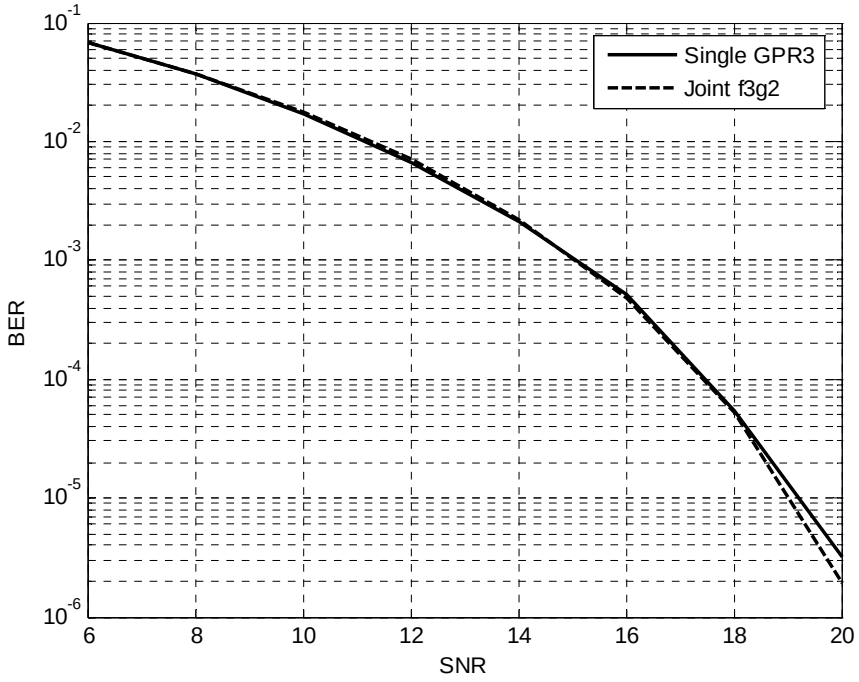


Fig. 7.4. Performance of single-track and joint-track equalized BPMR channels.

To understand this phenomenon, let us re-consider the relation between equalization methods and the associated detection. The single-track equalization minimizes the mean square error $\mathrm{MSE}_{\mathrm{SE}} = E\{e_k^2\}$, with $e_k = z_k - \sum_l f_l a_{k-l}$, which is with respect to a_k. Since a_k is the desired data, $\mathrm{MSE}_{\mathrm{SE}}$ should be a good proxy metric for BER performance of the

single-track equalization. The joint-track equalization minimizes the mean square error $\mathrm{MSE}_{\mathrm{JE}} = E\{e_k^2\}$, with $e_k = z_k - \sum_l f_l a_{k-l} - \sum_l g_l d_{k-l}$, where both a_k and d_k are treated as desired sequences, and hence $\mathrm{MSE}_{\mathrm{JE}}$ is with respect to a_k and d_k. Since the joint-track equalization utilizes the distribution of d_k on {–2 , 0, +2}, it can achieve a smaller MSE than the single-track equalization, i.e., $\mathrm{MSE}_{\mathrm{JE}} < \mathrm{MSE}_{\mathrm{SE}}$. However, during the detection on the 2D trellis, the APPs $\mathrm{P}(a_k, d_k \mid \mathbf{z})$ are computed and then marginalized to get the desired APPs for a_k, $\mathrm{P}(a_k \mid \mathbf{z})$. Thus, d_k is indeed not the desired data. In the process, the detection of d_k increases the uncertainty of the detection of a_k , just like e_k which is assumed to have a Gaussian distribution during detection. Therefore, it is appropriate to look at $\sum_l g_l d_{k-l}$ as an additional noise. Since $\mathrm{MSE}_{\mathrm{JE}}$ is the MSE with respect to both a_k and d_k, it obviously is not a good metric for detection performance. Instead, the detection performance of a_k should be related to the MSE with respect to a_k. In other words, what the joint-track equalization minimizes is not exactly the quantity desired by the detector.

We define the MSE with respect to a_k as the effective MSE (EMSE). For the joint-track equalization, $\mathrm{EMSE}_{\mathrm{JE}} = E\{(e_k')^2\}$, where $e_k' = z_k - \sum_l f_l a_{k-l} = e_k + \sum_l g_l d_{k-l}$. The effective MSE is computed as

$$\mathrm{EMSE}_{\mathrm{JE}} = \mathbf{f}^{\mathrm{T}}\mathbf{R}_{\mathbf{a}}\mathbf{f} + \mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y},\mathbf{a}}\mathbf{f}, \tag{7.14}$$

which has exactly the same expression as (7.2), the MSE of the single-track equalization; but (7.14) is not the quantity to be minimized in joint-track equalization. Note that, from the definition of EMSE, it is clear that $\mathrm{EMSE}_{\mathrm{JE}} \geq \mathrm{MSE}_{\mathrm{JE}}$ and $\mathrm{EMSE}_{\mathrm{SE}} = \mathrm{MSE}_{\mathrm{SE}}$.

We show in Fig. 7.5 the MSEs and EMSEs for single-track and joint-track equalized BPMR channels simulated in this section. We see that joint-track equalization has much smaller MSE but larger EMSE than single-track equalization. We argue that it is the large

$EMSE_{JE}$ that causes the poor performance of the joint-track detection, which was expected to be better, given the small MSE_{JE}.

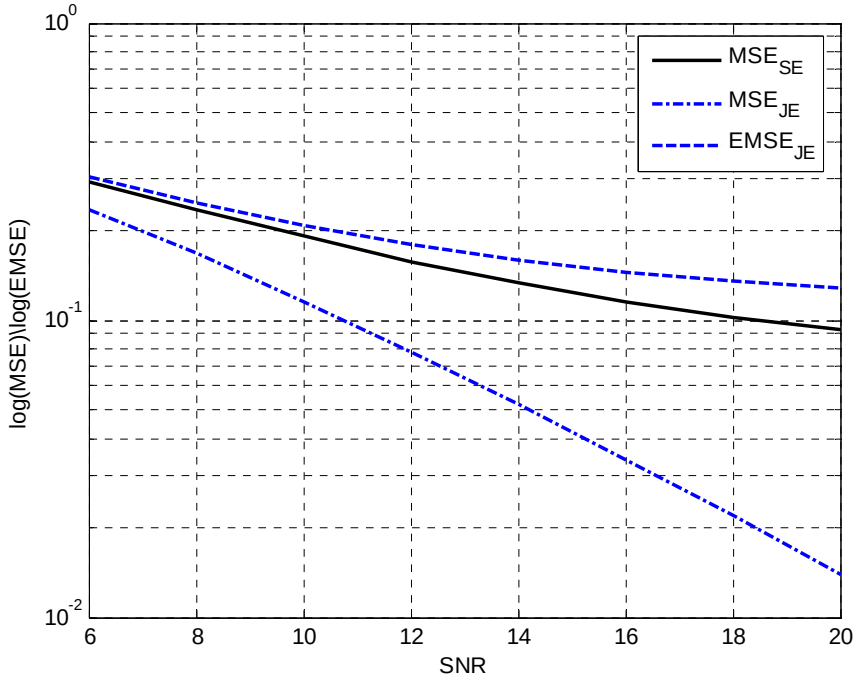


Fig. 7.5. Mean-squared errors of single-track and joint-track equalizations.

However, if the performance of the joint-track detection is related to the $EMSE_{JE}$, how can we explain the similar performance for the two equalization methods with $EMSE_{JE} \geq MSE_{SE}$? We think that joint-track equalization definitely has an advantage over single-track equalization. In [10], a simple joint-track model was investigated, which is shown in Fig. 7.6, where n_k is AWGN; **f** and **g** are FIR filters with $\mathbf{g} = \alpha\mathbf{f}$ and α is a scalar (attenuation factor). The signal at the input of the detector can be written as

$$z_k = \sum_l f_l a_{k-l} + \sum_l g_l b_{k-l} + n_k \,. \tag{7.15}$$

If we treat $\sum_l g_l b_{k-l} + n_k$ as noise and do detection on the trellis constructed with target **f**, then the model shown in Fig. 7.6 is a single-track model with ITI. Otherwise, if n_k is

considered the only noise and the detection is running on the trellis based on a 2D target of **f** and **g**, the model shown in Fig. 7.6 becomes a joint-track model. Given that the attenuation factor $0 < \alpha \leq 0.2$, it has been proved that the information rate of the joint-track model is greater than that of the single-track model, where all information rates are with respect to a_k. We note that the single-track model has the same MSE with respect to a_k as the joint-track model, which is equivalent to the case $MSE_{SE} = EMSE_{JE}$. It is well known that it is necessary to have a powerful error correcting code to achieve the information rate; the information rate may not be a good indicator of BER performance for the uncoded channel. However, the better information rate offers the possibility of improving the BER performance of the uncoded channel.

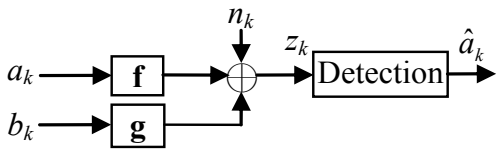


Fig. 7.6. A simple channel model for the investigation of single-track and joint-track equalizations.

In summary, the performance of the joint-track equalized BPMR channel cannot be simply predicted by MSE_{JE} or $EMSE_{JE}$. The large $EMSE_{JE}$ negatively impacts the detection performance, while joint-track equalization has the advantage on information rate. We believe a compromise between these two points leads to the joint-track equalized channel exhibiting a similar performance to the single-track equalized case.

7.4 Multi-track detection

According to our analysis, joint-track equalization provides much smaller MSE but

larger EMSE. Is it possible to reduce the impact of $EMSE_{JE}$ and take advantage of the small MSE_{JE}? We note that if the data on the side tracks were perfectly known, then the noise e_k' has a mean of $\sum_l g_l d_{k-l}$, and its variance $\text{var}\{e_k'\} = E\{e_k^2\} = MSE_{JE}$, i.e., the noise variance with respect to a_k reduces to MSE_{JE}. Since MSE_{JE} is much smaller than MSE_{SE}, we expect that joint-track equalization with known d_k will provide a significant gain over single-track equalization. Of course, in an actual channel, the data on the side tracks are usually unknown; but we think that having some *a priori* information about the data on the side tracks during joint-track detection is still equivalent to reducing the noise variance with respect to a_k from $EMSE_{JE}$ to some smaller value which is greater than MSE_{JE}.

Therefore, we propose a multi-track detection technique. The basic idea is that, before we detect the center track, the two side tracks will be detected first and the APPs of b_k and c_k are obtained, which we use in turn as the *a priori* information during detection on the center track. To further clarify the idea, we give the equations below to take into account the *a priori* information of b_k and c_k in the joint-BCJR detection on the $\{a_k, d_k\}$ trellis. Following the discussion in Section 7.3.2, and given that the detector is implemented in the logarithm domain, we have

$$\log(P(a_k = -1, d_k = -2)) = \text{A},$$

$$\log(P(a_k = -1, d_k = 0)) = \log\left(\exp(L_{bk}) + \exp(L_{ck})\right) + \text{A},$$

$$\log(P(a_k = -1, d_k = 2)) = L_{bk} + L_{ck} + \text{A},$$

$$\log(P(a_k = 1, d_k = -2)) = L_{ak} + \text{A},$$

$$\log(P(a_k = 1, d_k = 0)) = L_{ak} + \log\left(\exp(L_{bk}) + \exp(L_{ck})\right) + \text{A},$$

$$\log(P(a_k = 1, d_k = 2)) = L_{ak} + L_{bk} + L_{ck} + \text{A},$$

where L_{ak}, L_{bk} and L_{ck} are log-likelihood ratios (LLRs) with $L_{ak} = \log(P(a_k = 1) / P(a_k =$

-1)) and similarly for L_{bk} and L_{ck}; A = $-$ log(1+exp(L_{ak})) $-$ log(1+exp(L_{bk})) $-$ log(1+exp(L_{ck})) is a constant for different combinations of $\{a_k, d_k\}$ and hence could be ignored in the detection. Note that there is still no *a priori* information for a_k, i.e., $L_{ak} = 0$, during center track detection.

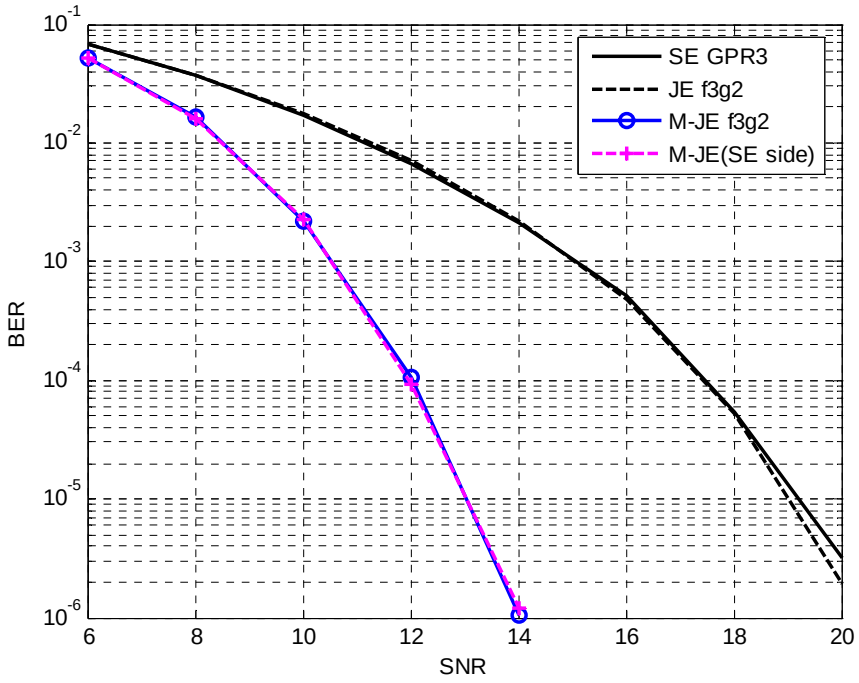


Fig. 7.7. Performance of multi-track detection with joint-track equalization.

We give the BER performance of the multi-track detection on the joint-track equalized BPMR channel in Fig. 7.7, where we achieve a gain of about 6 dB over the single-track equalized channel at the BER of 10^{-5}, which is consistent with our analysis. In our simulation, we assume that sectors on all three tracks are synchronized, i.e., sectors always begin and end at the same time ticks on all tracks. Indeed, this assumption is not necessary; but the sector offset must be known if there is any. In addition, to apply multi-track detection, only the center track needs to be equalized to a 2D PR target; it is acceptable to use any other equalization and detection method on the side tracks, if a different equalizer

and detector can be implemented in the same system. For example, we can use single-track equalization and use the standard BCJR detection on the side tracks, while the center track remains joint-track equalized. The performance of such a hybrid system is also shown in Fig. 7.7, where the hybrid system clearly provides almost the same performance as the joint-track equalized channel with multi-track detection, but has lower computational complexity.

Furthermore, it is obvious that the detection of side tracks introduces a delay in the system. But we think that, in practice, a forward only detection algorithm can be used on the side tracks or even on all three tracks to allow the parallel implementation of detection on all tracks.

Before we conclude this section, we need to clarify our terminology to avoid confusion. First, detecting a track means we are running a detection algorithm and generate an estimate of the data on that track; any sensing or even reading back signals from a track is not considered detection, unless there is a detector running on that track. Second, the multi-track detection introduced in this paper aims at data recovery on the center track only; all estimates of the data on side tracks are discarded. In other words, we are not using this technique to recover data from multiple tracks simultaneously.

7.5 2D equalization

As we have emphasized, the key point is that the center track must be equalized to a 2D GPR target, which allows us to capitalize on the smaller MSE. Actually, besides the joint-track equalization, there is another way to equalize read channels to 2D GPR targets, namely a 2D equalization technique [74]. The 2D equalization technique employs a 2D

equalizer to mitigate the impact of ITI, by equalizing the readback signals from multiple tracks. To avoid the use of complicated detection algorithms, the GPR targets for 2D equalization in [74] are constrained to be 1D. 2D equalization with a 1D GPR target, denoted by 2D1D equalization, provides significant gains over the single-track equalization on high-density BPMR channels. However, with a 2D GPR target, the 2D equalization technique is expected to give an MSE (denoted as MSE_{2D2D}) smaller than MSE_{2D1D}. If it is true, then we may use multi-track detection to achieve additional gains, just like what we did for the joint-track equalization case.

7.5.1 BPMR channel with 2D equalization

The BPMR channel with 2D equalization is shown in Fig. 7.8, where five tracks {–2, –1, 0, 1, 2} have been sensed; three heads read back the signal on three tracks {–1, 0, 1}; the data on the center track, $a_{0,k}$, are detected. The equalizers on the three tracks are $\mathbf{w}_{-1} = [w_{-1,-N}, \ldots, w_{-1,0}, \ldots, w_{-1,N}]^{\mathrm{T}}$, $\mathbf{w}_0 = [w_{0,-N}, \ldots, w_{0,0}, \ldots, w_{0,N}]^{\mathrm{T}}$ and $\mathbf{w}_1 = [w_{1,-N}, \ldots, w_{1,0}, \ldots, w_{1,N}]^{\mathrm{T}}$, where we take $N = 10$ as in other equalization methods. The PR targets for the three tracks are $\mathbf{t}_{-1} = [t_{-1,0}, \ldots, t_{-1,L2-1}]^{\mathrm{T}}$, $\mathbf{t}_0 = [t_{0,0}, \ldots, t_{0,L1-1}]^{\mathrm{T}}$ and $\mathbf{t}_1 = [t_{1,0}, \ldots, t_{1,L2-1}]^{\mathrm{T}}$. Define vectors, $\mathbf{w} = [\mathbf{w}_{-1}{}^{\mathrm{T}} \ \mathbf{w}_0{}^{\mathrm{T}} \ \mathbf{w}_1{}^{\mathrm{T}}]^{\mathrm{T}}$, $\mathbf{t} = [\mathbf{t}_{-1}{}^{\mathrm{T}} \ \mathbf{t}_0{}^{\mathrm{T}} \ \mathbf{t}_1{}^{\mathrm{T}}]^{\mathrm{T}}$; $\mathbf{y}_{-1} = [y_{-1,k+N}, \ldots, y_{-1,k-N}]^{\mathrm{T}}$, $\mathbf{y}_0 = [y_{0,k+N}, \ldots, y_{0,k-N}]^{\mathrm{T}}$, $\mathbf{y}_1 = [y_{1,k+N}, \ldots, y_{1,k-N}]^{\mathrm{T}}$ and $\mathbf{y} = [\mathbf{y}_{-1}{}^{\mathrm{T}} \ \mathbf{y}_0{}^{\mathrm{T}} \ \mathbf{y}_1{}^{\mathrm{T}}]^{\mathrm{T}}$; $\mathbf{a}_{-1} = [a_{-1,k}, \ldots, a_{-1,k-L2+1}]^{\mathrm{T}}$, $\mathbf{a}_0 = [a_{0,k}, \ldots, a_{0,k-L1+1}]^{\mathrm{T}}$, $\mathbf{a}_1 = [a_{1,k}, \ldots, a_{1,k-L2+1}]^{\mathrm{T}}$ and $\mathbf{a} = [\mathbf{a}_{-1}{}^{\mathrm{T}} \ \mathbf{a}_0{}^{\mathrm{T}} \ \mathbf{a}_1{}^{\mathrm{T}}]^{\mathrm{T}}$. Then the equalization error e_k can be expressed as

$$e_k = \mathbf{w}^{\mathrm{T}}\mathbf{y} - \mathbf{t}^{\mathrm{T}}\mathbf{a}. \tag{7.16}$$

With $\mathbf{R}_{\mathbf{y}} = E\{\mathbf{y}\mathbf{y}^{\mathrm{T}}\}$, $\mathbf{R}_{\mathbf{a}} = E\{\mathbf{a}\mathbf{a}^{\mathrm{T}}\}$ and $\mathbf{R}_{\mathbf{y},\mathbf{a}} = E\{\mathbf{y}\mathbf{a}^{\mathrm{T}}\}$, the mean-squared error can be expressed as

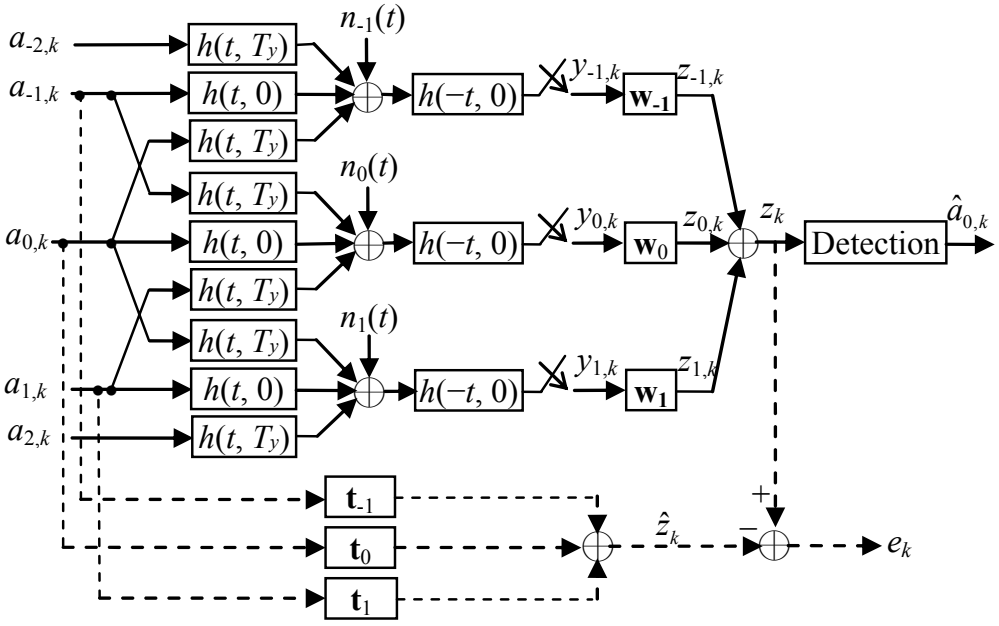


Fig. 7.8 A 2D equalized BPMR channel model.

$$\mathrm{MSE}_{2\mathrm{D}} = E\left\{e_k^2\right\} = \mathbf{t}^{\mathrm{T}}\mathbf{R}_{\mathbf{a}}\mathbf{t} + \mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y,a}}\mathbf{t}\,. \tag{7.17}$$

By minimizing (7.17) and enforcing $t_{0,0} = 1$, the optimized GPR target and equalizer can be computed by

$$\mathbf{t} = \lambda(\mathbf{R}_{\mathbf{a}} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R}_{\mathbf{y,a}})^{-1}\mathbf{C}\,, \tag{7.18}$$

$$\mathbf{w} = \mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R}_{\mathbf{y,a}}\mathbf{t}\,, \tag{7.19}$$

where $\mathbf{C} = [0 \ldots 0,1,0 \ldots 0]^{\mathrm{T}}$ is a vector of length $L_1 + 2L_2$, and

$$\lambda = \frac{1}{\mathbf{C}^{\mathrm{T}}(\mathbf{R}_{\mathrm{a}} - \mathbf{R}_{\mathbf{y,a}}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}^{-1}\mathbf{R}_{\mathbf{y,a}})^{-1}\mathbf{C}}.$$

Since we use 1D vectors to express 2D signals and filters, the design of 2D equalization follows the same method of the single-track equalization; (7.17)−(7.19) resemble (7.2)−(7.4). In [74], the 2D equalizer, 2D target and the read back signals from three tracks are expressed as a matrix and the filter input-output relations are written as 2D

convolutions, which give a more compact description of the system. However, our decomposed expressions and derivations provide a better understanding of the 2D equalized BPMR channel. In addition, in [74], to get different target lengths for the center and side tracks or even 1D target, a special constraint matrix E needs to be constructed and used in the minimization of MSE_{2D} to force some target coefficients to zero. But our interpretation of the 2D equalization technique naturally avoids such a problem; simple adjustment on L_1 and L_2 can implement targets with different lengths; to design a 1D target, all we have to do is to set $\mathbf{t} = \mathbf{t}_0$ and $\mathbf{a} = \mathbf{a}_0$. As in the joint-track equalization, the symmetry of the channel model makes the same equalizers and targets on side tracks numbered -1 and 1, i.e., $\mathbf{w}_{-1} = \mathbf{w}_1$ and $\mathbf{t}_{-1} = \mathbf{t}_1$, and hence the detection can be on the trellis with inputs $\{a_{0,k}, a_{-1,k} + a_{1,k}\}$. In addition, the additive white Gaussian noises $n_{-1}(t)$, $n_0(t)$ and $n_1(t)$ are assumed to be independent of each other and have the same double sided power density height of σ^2. We define the SNR as $1/\sigma^2$, which is consistent with the SNR definition in the single-track and joint-track equalized channels.

7.5.2 Performance of 2D equalized BPMR channels

We simulate the 2D equalized channels on the same medium used in Section 7.3 and 7.4. In the 2D1D equalization, the channel is equalized to GPR3 targets, while in the 2D2D equalization, we choose L_1=3 and L_2=2; longer PR targets for side tracks give only negligible improvement on $MMSE_{2D2D}$ and the BER performance. We show the simulation results in Fig. 7.9, where the 2D1D equalized channel exhibits an excellent performance, with about a 5-dB gain over the single-track equalized channel at a BER of 10^{-5}, which is only 1 dB worse than the joint-track equalized channel with multi-track detection. However, the 2D2D equalized channel has poor performance, which is even worse than the

single-track equalized channel at low SNRs. This time, this is not surprising. Let us take a look at the EMSE of the 2D2D equalizer, which is computed by

$$\mathrm{EMSE}_{2D2D} = \mathbf{t}_0^{\mathrm{T}}\mathbf{R}_{\mathbf{a0}}\mathbf{t}_0 + \mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y}}\mathbf{w} - 2\mathbf{w}^{\mathrm{T}}\mathbf{R}_{\mathbf{y,a0}}\mathbf{t}_0, \tag{7.20}$$

where $\mathbf{R}_{\mathbf{a0}} = E\{\mathbf{a}_0\mathbf{a}_0^{\mathrm{T}}\}$ and $\mathbf{R}_{\mathbf{y,a0}} = E\{\mathbf{y}\mathbf{a}_0^{\mathrm{T}}\}$. We show the MSEs of different equalizations in Fig. 7.10. First, the MSE_{2D1D} is much smaller than MSE_{SE}, which explains the large gain achieved by 2D1D equalization over single-track equalization. Second, the EMSE_{2D2D} is the worst among all equalization methods, which could be related to the poor performance of the 2D2D equalized channel. Finally, MSE_{2D2D} is even slightly smaller than MSE_{JE}, which lets us expect an additional gain by using multi-track detection with 2D2D equalization.

7.5.3 2D equalization with multi-track detection

We give the BER performance of multi-track detection on the 2D2D equalized BPMR channel in Fig. 7.11, where the 2D2D equalized channel aided by multi-track detection outperforms the 2D1D equalized channel, but is still worse than the joint-track equalized channel with multi-track detection. Given that $\mathrm{MSE}_{2D2D} \leq \mathrm{MSE}_{JE}$ in Fig. 7.10, we still expect that 2D2D equalization can perform better. We notice that 2D2D equalization has the worst performance in Fig. 7.9, which means that in the multi-track detection of the 2D2D equalized channel, the APPs of the data on the side tracks may not be reliable. Just as we discussed in Section 7.4, it is possible to employ different equalization methods on the side tracks. Given that 2D1D equalization provides very good performance, we choose this method to equalize the side tracks and keep the 2D2D equalization for the center track. We present the simulation results in Fig. 7.11, where this hybrid 2D equalization system

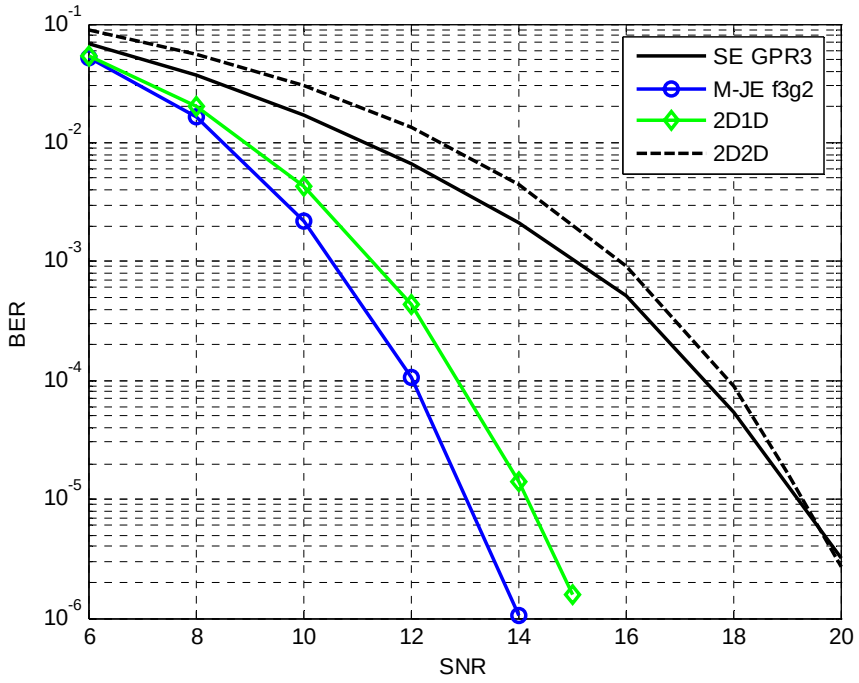


Fig. 7.9. Performance of the 2D equalized BPMR channels.

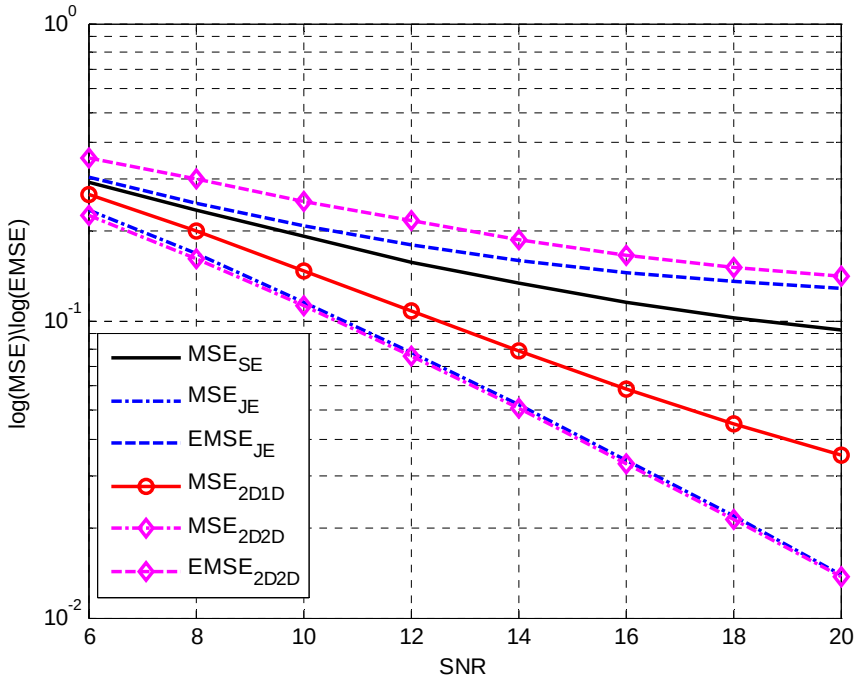


Fig. 7.10. Mean-squared errors for three different equalizations.

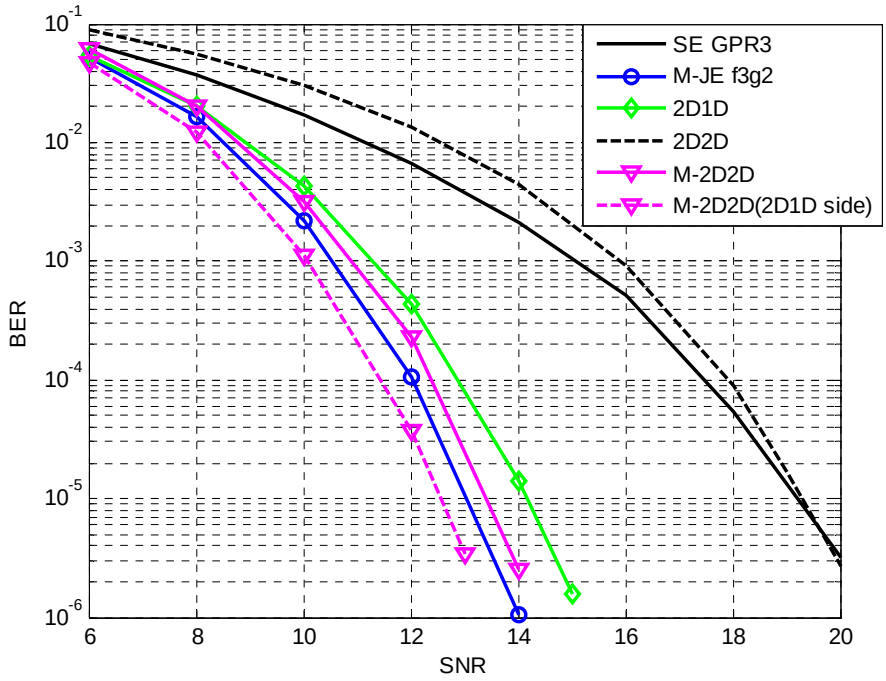


Fig. 7.11. Performance of the 2D equalized BPMR channels with multi-track detection.

aided by multi-track detection gives the best performance.

7.6 Performance bounds for multi-track detection techniques

As we discussed in Section 7.4, multi-track detection provides probability information for the data on side tracks to be used for center track detection, which is considered equivalent to bringing the EMSE closer to the MSE. On the other hand, better BER performance on side track detection improves the detection performance on the center track. If the detection on side tracks is error free or equivalently, the data on the side tracks are known, then EMSE = MSE and the detection on the center track will have the best performance, which can be treated as a performance bound for multi-track detection techniques.

We have introduced the detection on the trellis of a 2D PR target in Section 7.3.2.

However, given that the data on the side tracks are known, the trellis and the detection method are much simpler. On the trellis with input $\{a_k, d_k\}$, if d_k is known, the trellis will have the same number of states and state transition relations as the one constructed on the 1D target for a_k; the contribution of d_k and its PR target remains only in the computation of the branch values.

We do the simulation on the same medium as in previous sections. We show the performance bounds for joint-track and 2D2D equalized channels in Fig. 7.12, where the two performance bounds are very close, which is reasonable due to the small difference between $\mathrm{MSE}_{\mathrm{JE}}$ and $\mathrm{MSE}_{\mathrm{2D2D}}$. It is interesting to note that the hybrid 2D equalization with multi-track detection approaches the performance bound at medium and high SNRs, while the joint-track equalization with multi-track detection does not. Obviously, the joint-track equalization with multi-track detection needs a better detection performance on the side tracks to reach the performance bound. An available choice is 2D1D equalization for the side tracks, which is expected to lead to another high performance hybrid system, just like the hybrid 2D equalization proposed in Section 7.5.3. However, there may be some difficulties in implementing different equalizers in one system. To avoid constructing hybrid systems, we consider a simple extension of the multi-track detection technique.

To improve the detection performance on side tracks, we can apply the multi-track detection technique on the two side tracks of the center track, from which we are retrieving data; then the tracks adjacent to each side track need to be detected first. Given that the center track is numbered 0, and other tracks are numbered in order as {…, −2, −1, 0, 1, 2, …}, the tracks −2, 0 and 2 are detected at first; then the detection on tracks −1 and 1 will be aided by the probability information from tracks −2, 0 and 2. In this way, the detection

performance on tracks −1 and 1 will be improved, and hence the BER performance of the center track (track 0) is expected to achieve the performance bound. Note that there are a total of five tracks, which have been detected. This extension of the multi-track detection technique only requires one equalization method in the system and can be applied with both joint-track and 2D equalizations.

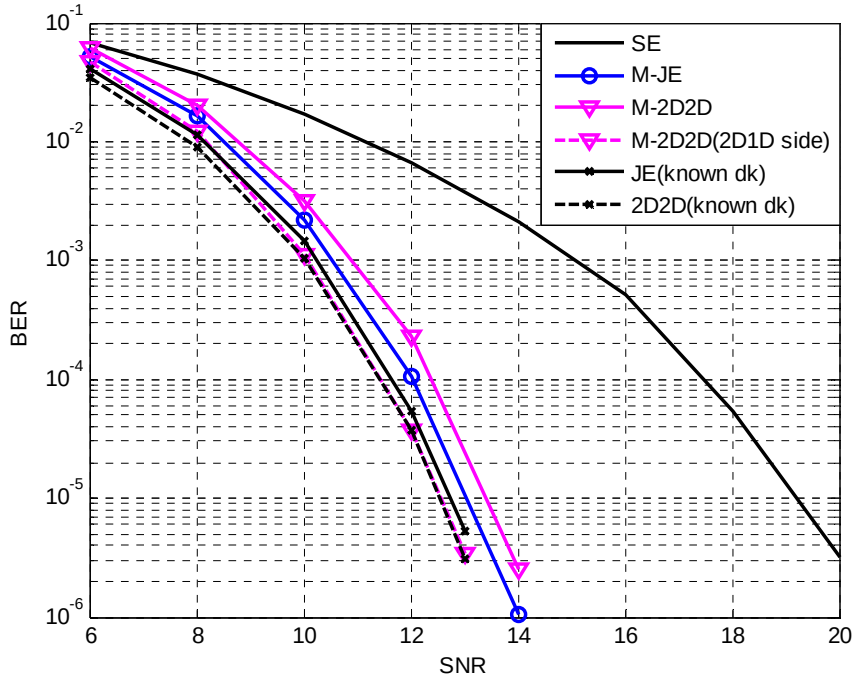


Fig. 7.12. Performance bounds for joint-track and 2D2D equalization.

We present in Fig. 7.13 the performance of the hybrid system with joint-track equalized center track and 2D1D equalized side tracks, as well as the performance of the extended multi-track detection with five joint-track equalized tracks. Clearly, both of them achieve the performance bound. We apply the extended multi-track detection technique to the 2D equalized channel; we show the simulation results in Fig. 7.14, where the performance bound is also achieved.

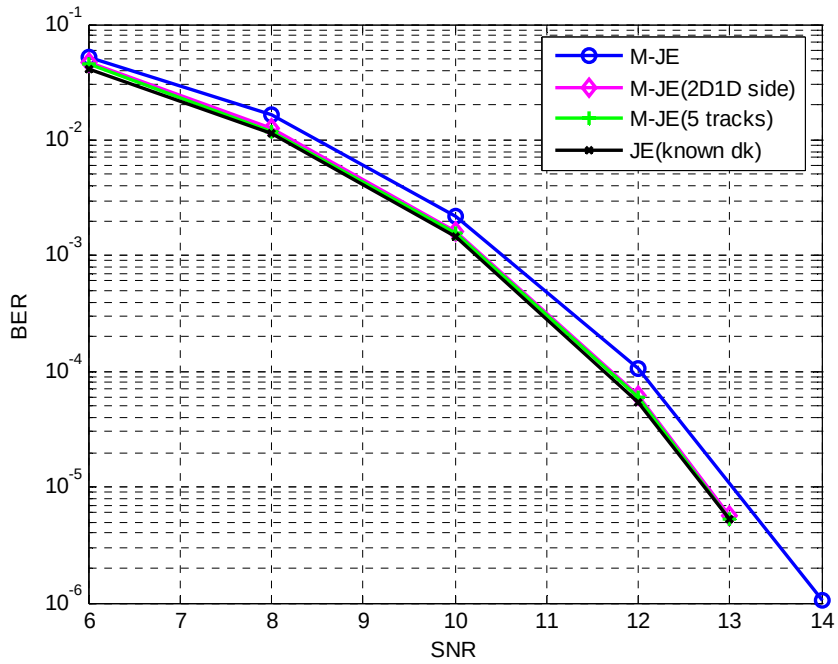


Fig. 7.13. Achieving the performance bound for multi-track detection with joint-track equalization.

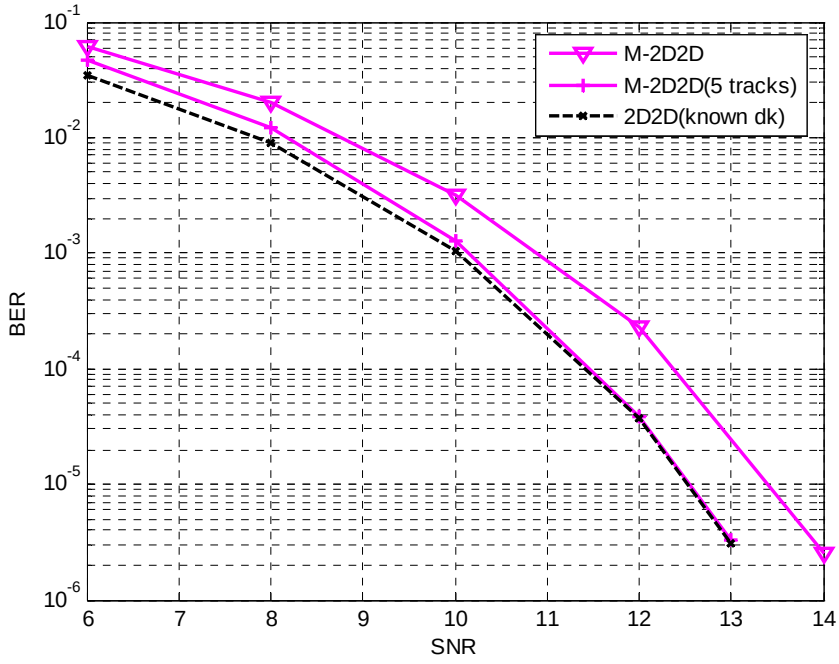


Fig. 7.14. Achieving the performance bound for multi-track detection with 2D equalization.

7.7 Simulation with other island distributions

In previous sections, we have investigated the proposed techniques on a bit-patterned medium which is characterized by strong ISI and ITI (PW_N = 1.5 and A_s = 0.2). In this section, we provide simulations on the other two typical media to further validate the analysis presented.

First, we consider a bit-patterned medium with strong ISI and weak ITI. We choose an arrangement of islands with T_x = 13 nm, T_y = 24.7 nm, which gives an areal density around 2Tb/in^2, where the normalized pulse width is PW_N = 1.5 and the side track amplitude is A_s = 0.0625. The simulation results are shown in Fig. 7.15, where the performance bounds only outperform the single-track equalized channel by around 1.3 dB; the simple 2D1D equalization harvests most of the gain. The multi-track detection technique was proposed to mitigate ITI; it is to be expected that this technique is not effective on the channel with weak ITI.

Second, we consider a bit-patterned medium, in which the islands are squarely distributed with T_x = T_y = 18 nm. This medium gives an areal density also around 2Tb/in^2 and the related read channel has weak ISI and strong ITI (PW_N = 1.083 and A_s = 0.2294). We show the simulation results in Fig. 7.16, where the multi-track detection with both joint-track and 2D equalizations, provide significant gains over the single-track equalization. However, we see that there are still visible gaps between the performance bounds and the best performance given by the hybrid systems. Again, we use the extended multi-track detection to obtain better detection performance on the two side tracks. We present the simulation results in Fig. 7.17, where the performance bounds are finally achieved.

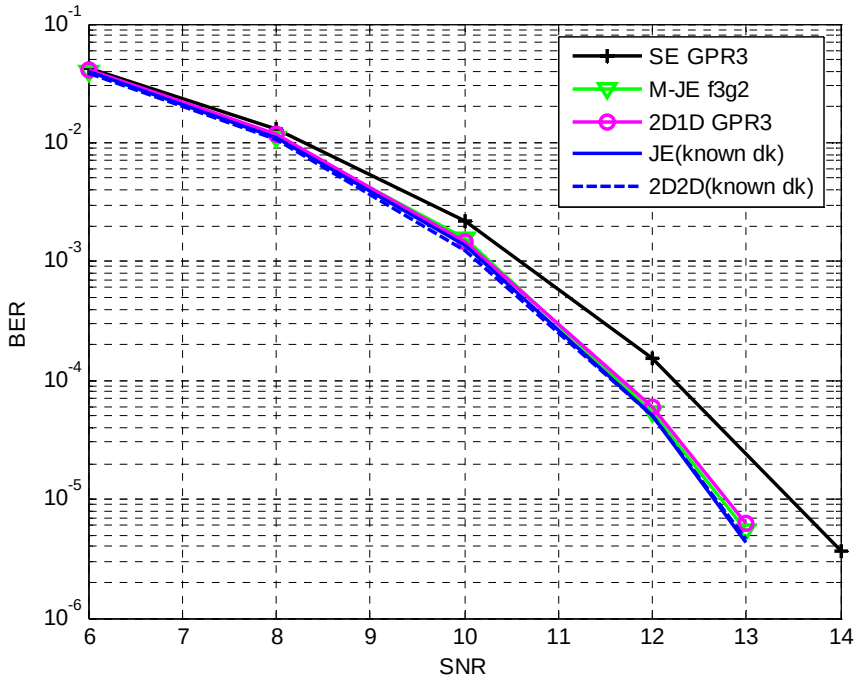


Fig. 7.15. Simulations on BPMR channels with strong ISI and weak ITI.

7.8 Conclusion

In this chapter, we provide new insights into equalization methods with 2D GPR targets, and propose multi-track detection techniques that can achieve the performance bounds. Simulations on bit-patterned media of three typical island distributions fully validate our analysis. Finally we point out that multi-track detection complicates the implementation of a practical system and introduces time delay during user data retrieval. We argue that using forward-only detection on the side tracks (or even on all tracks) makes it possible to do detection on all tracks in parallel and hence avoiding the time delay. In addition, joint-track equalization with multi-track detection has similar performance as 2D equalization, with fewer track readings and hence lower complexity.

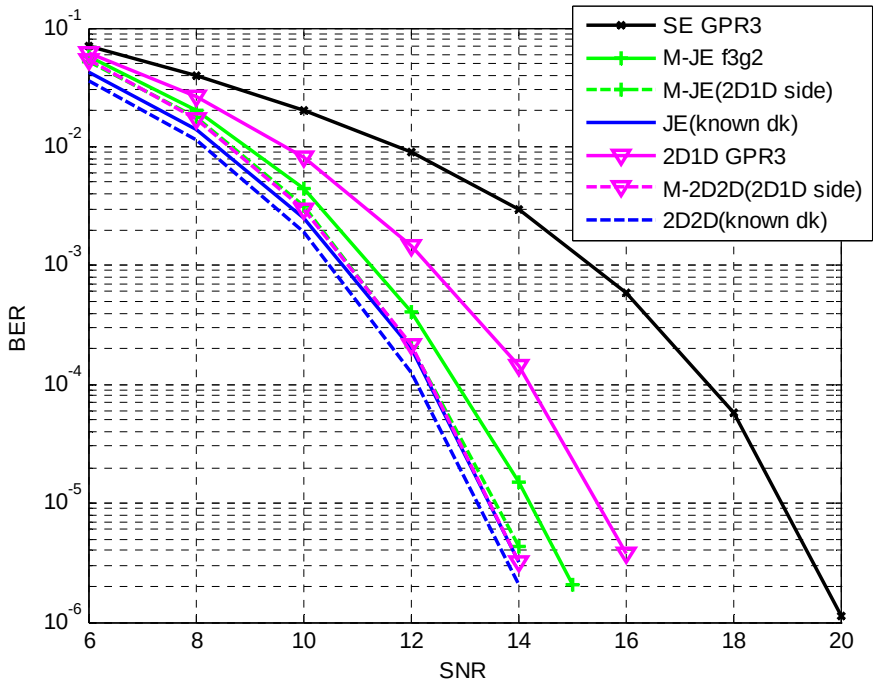


Fig. 7.16. Simulations on BPMR channels with weak ISI and strong ITI.

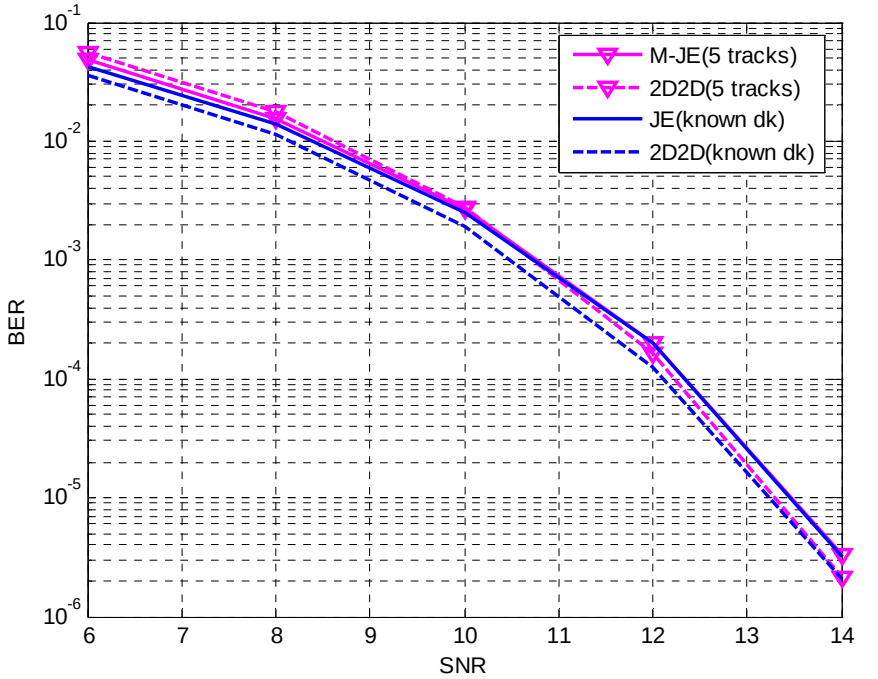


Fig. 7.17. Achieving the performance bound on BPMR channels with weak ISI and strong ITI.

8 Epilogue

In this dissertation, we investigated advanced signal processing techniques for magnetic recording channels, and proposed novel methods for channel detection, LDPC code design and decoding. In this chapter, we conclude the investigation, outline the contributions in this dissertation and give suggestions for future work.

8.1 Conclusions

At the beginning of this dissertation, we clearly introduced the basics of modern magnetic recording systems. The equalization method and an appropriate SNR definition given in Chapter 1 are employed in the investigation of PMRCs in Chapters 3−6, which make the results of our simulations comparable across chapters. In addition, we also introduced a new technique for the next generation of magnetic recording system, investigated the new bit-patterned media and replay response and outlined the challenges of this novel recording technique. Then we made the following contributions on both PMRCs and BPMR channels in Chapters 3−7.

1) A symbol-based BCJR algorithm is proposed in Chapter 3, by which we can implement turbo equalization for nonbinary LDPC coded PR channels exactly.

   Furthermore, the simplified versions of the symbol-based BCJR algorithm are derived for PR channels and their computational complexities are calculated and compared. The simulations on PMRCs show that the proposed algorithm significantly outperforms the conventional methods.

2) An improved BP (IBP) decoder for LDPC coded PR channels is proposed in Chapter 4, which makes use of the correlations between the channel messages in the BP decoding.

The IBP decoder is thoroughly investigated on LDPC coded PR channels; it turned out to achieve better performance with stronger correlations between channel messages. The simulations on PMRCs further validate this investigation. Without the turbo equalization, the IBP decoder provides very large gains over the standard BP decoder. Additional research shows that the turbo iterations do not help the IBP decoder very much but significantly improve the performance of LDPC coded channels with the standard BP decoder. We conjecture that the standard BP decoder may obtain and use the correlations between channel messages by turbo iterations; our algorithm provides another way to harvest the gain. Finally, the IBP decoder is further extended for the nonbinary LDPC coded PR channels.

3) In Chapter 5, we introduced several new methods to design LDPC codes for magnetic recording, where we are aiming at reducing the number of short cycles on the factor graphs of LDPC codes.

First, a modified PEG (MPEG) algorithm is proposed to do the LDPC code construction. Then we applied the MPEG algorithm and the lattice construction technique to the design of QC-LDPC codes. QC-LDPC codes with fewer shortest cycles were obtained. Meanwhile, we find that only reducing the number of the shortest cycles may boost the number of the longer cycles in the code, which may offset the performance improvement. Finally, we point out

that the MPEG algorithm itself not only eliminates the shortest cycles but also reduces the number of larger short cycles in a more general sense, and hence always gives LDPC codes with good performance.

4) In Chapter 6, we made a substantial investigation of the RS plus LDPC coded PMRCs, which is presently used in commercial implementations.

   After designing a group of inner LDPC codes with various code rates and column weights, we evaluate the SER performance of the concatenated codes with different iterative schemes. Given a fairly good iterative scheme, we find the optimal code rates of the concatenated codes by simulation. Then the contributions of the outer RS codes are validated by simulations in both random noise and media defects. In addition, we utilized a recently developed technique, the microscopic method to estimate the error floors of the concatenated codes whose inner LDPC codes have column weight of two.

5) Advanced equalization and detection methods for BPMR channels are investigated in Chapter 7, where we proposed a multi-track detection technique to mitigate the ITI.

   The proposed technique can take full advantage of equalization methods with 2D GPR targets. The multi-track detection with both joint-track or 2D equalization provides significant performance improvement compared to conventional equalization and detection methods. Furthermore, various detection strategies based on the multi-track detection, including several hybrid detection methods and the extended multi-track detection technique, are introduced to achieve the performance bounds.

8.2 Future work

Based on the work carried out in this dissertation, we list some possible issues which could be addressed in future.

- The IBP decoder introduced in Chapter 4 has very good performance. But in practice, noise predictive detectors are used in magnetic recording systems. Although we have mentioned that the advantage of the IBP decoder still holds with noise prediction, the performance of IBP needs to be re-evaluated. Note that designing the noise-predictive channel detector for IBP decoding is also a challenge.
- The IBP decoder has higher computational complexity than the standard BP decoder. But by using the forward-only channel detector and a simplified decoding algorithm such as Min-Sum, it is possible to design a reduced complexity decoder based on IBP. As we see in Section 4.6, although the IBP decoder has a comparable performance with the standard BP decoder with turbo equalization, the IBP decoder can run faster than the standard BP decoder, because no turbo iterations are needed for IBP. Therefore, it is interesting to compare the performance and complexities of the IBP and BP decoders with a more practical magnetic recording system.
- In Chapter 6, we estimated the error floors of RS plus LDPC coded PMRCs, where the inner codes have column weight of two. But it would be interesting to find the error floors or estimate the performance at high SNRs for codes with higher column weights. In addition, it is also important to estimate the performance of LDPC-only coded MRCs at high SNR, which will make the

investigation in Chapter 6 more meaningful and attractive. However, the error rate estimation is still a big challenge for LDPC coded MRCs.

- BPMR for high recording density is a new recording technique. Before it is implemented in products, there are still a lot of BPM-specific challenges that need to be overcome. What we did in Chapter 7 is only the beginning of this work, where only the ISI, ITI and electronic noise are included in the channel; written-in errors and media noises such as island location and size fluctuations also need to be considered in the future. In addition, with extremely high recording density, the ITI may be caused by more than two side tracks, which will make the multi-track detection more complicated to implement. One way to reduce the complexity is keeping the 2D target for only the center and the nearest two side tracks, but the performance of this technique need to be re-investigated.
- To mitigate the ITI in BPMR channels, we considered to re-design the equalizer and channel detectors. But it is possible to reduce the impairment of high density BPMR channels by error correcting coding. Although we have not done any work in this area, we are aware of that it is an important topic and some progress has been made by other researchers [78]-[80].

Bibliography

[1] S. X. Wang, A. M. Taratorin, *Magnetic Information Storage Technology*, San Diego/London/Boston/New York/Sydney/Tokyo/Toronto: Academic Press, 1998.

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

[3] S. Khizroev and D. Litvinov, *Perpendicular Magnetic Recording*, Boston/Dordrecht/London: Kluwer Academic Publishers, 2004.

[4] R. Rottmeyer *et al.*, "Heat-assisted magnetic recording," *IEEE Trans. Magn.*, vol. 42, no. 10, pp. 2417–2421, Oct. 2006.

[5] B. Terris, T. Thomson, and G. Hu, "Patterned media for future magnetic data storage," *Microsyst. Technol.*, vol. 13, no. 2, pp. 189–196, Nov. 2006.

[6] H. Sawaguchi, Y. Nishida, H. Takano, and H. Aoi, "Performance analysis of modified PRML channels for perpendicular recording systems," *J. Magnetism Magn. Materials*, vol. 235, pp. 265-272, Oct. 2001.

[7] Z. Wu, *Coding and Iterative Detection for Magnetic Recording Channels*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2000.

[8] J. Moon and W. Zeng, "Equalization for maximum likelihood detector," *IEEE Trans. Magn.*, vol. 31, no. 2, pp. 1083-1088, Mar. 1995.

[9] 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.

[10] W. Tan and J. R. Cruz, "Evaluation of detection algorithms for perpendicular recording channels with intertrack interference," *Journal of Magnetism and Magn. Materials,* vol. 287, pp. 397-404, 2005.

[11] X. Yang and E. M. Kurtas, "Signal and noise generation for magnetic recording channel simulations", in *Coding and Signal Processing for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas, Eds. Boca Raton: CRC Press, LLC, 2005, pp. 6-1-6-20.

[12] R. M. Todd, "A model for a perpendicular magnetic recording channel and equalizer," CSPLab, The University of Oklahoma, Tech. Rep., 2008.

[13] P. W. Nutter, D. McA. McKirdy, B. K. Middleton, D. T. Wilton, and H. A. Shute, "Effects of island geometry on the replay signal in patterned media storage," *IEEE Trans. Magn.,* vol. 40, no. 6, pp. 3551-3558, Nov. 2004.

[14] S. Nabavi, B. V. K. V. Kumar, and J. A. Bain, "Two-dimensional pulse response and media noise modeling for bit-patterned media," *IEEE Trans. Magn.,* vol. 44, no. 11, pp. 3789-3792, Nov. 2008.

[15] S. W. Yuan and H. N. Bertram, "Off-track spacing loss of shielded MR heads," *IEEE Trans. Magn.*, vol. 30, no. 3, pp. 1267–1273, May 1994.

[16] H. N. Bertram, *Theory of Magnetic Recording*, Cambridge, UK: Cambridge Univ. Press, 1994.

[17] H. A. Shute, D. T. Wilton, D. McA. McKirdy and D. J. Mapps, "Improved approximations for two-dimensional perpendicular magnetic recording heads," *IEEE Trans. Magn.*, vol. 39, no. 4, pp. 2098-2012, July 2003.

[18] D. T. Wilton, D. McA. McKirdy, H. A. Shute, J. J. Miles and D. J. Mapps, "Approximate three-dimensional head fields for perpendicular magnetic recording," *IEEE Trans. Magn.*, vol. 40, no. 1, pp. 148-156, Jan. 2004.

[19] H. A. Shute, D. T. Wilton, D. McA. McKirdy, P. M. Jermey and J. C. Mallinson, "Analytic three-dimensional response function of a double-shielded magnetoresistive or giant magnetoresistive perpendicular head," *IEEE Trans. Magn.*, vol. 42, no. 5, pp. 1611-1619, May 2006.

[20] S. Nabavi, "Signal processing for bit-patterned media channels with inter-track interference", Ph.D. dissertation, Dept. Elec. Engr. Comp. Science, Carnegie Mellon Univ., Pittsburgh, PA, 2008.

[21] K.Wiesen and B. Cross, "GMR head side-reading and bit aspect ratio," *IEEE Trans. Magn.*, vol. 39, no. 5, pp. 2609–2611, 2003.

[22] S. W. Yuan and H. N. Bertram, "Correction to "Off-track spacing loss of shielded MR heads"," *IEEE Trans. Magn.*, vol. 32, no. 4, July 1996.

[23] H. J. Richter,A.Y. Dobin, O. Heinonen, K. Z. Gao, R. J. M. V. D.Veerdonk, R. T. Lynch, J. Xue, D.Weller, P. Asselin, M. F. Erden, and R. M. Brockie, "Recording on bit-patterned media at densities of 1Tb/in^2 and beyond," *IEEE Trans. Magn.*, vol. 42, no. 10, Oct. 2006.

[24] J. Hu, T. M. Duman, E. M. Kurtas and M. F. Erden, "Bit-patterned media with written-in errors: modeling, detection, and theoretical limits," *IEEE Trans. Magn.*, vol. 43, no. 8, Aug. 2007.

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

[26] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Indust. Applied Mathematics*, vol. 8, pp. 300-304, 1960.

[27] J. Hagenauer and P. Hoeher, "A Viterbi algorithm with soft-decision outputs and its applications," in *Proc. IEEE Global Telecommun. Conf.*, 1989, pp. 1680-1686.

[28] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, pp. 284-287, Mar. 1974.

[29] A. Kavcic and J. M. F. Moura, "Correlation-sensitive adaptive sequence detection," *IEEE Trans. Magn.*, vol. 34, no. 3, pp. 763–771, May 1998.

[30] J. Moon and J. Park, "Pattern-dependent noise prediction in signal-dependent noise," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 730–743, Apr. 2001.

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

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

[33] A. Kavcic, X. Ma, and N. Varnica, "Matched information rate codes for partial response channels," *IEEE Trans. Inf. Theory*, vol. 51, no. 3, pp. 973-989, Mar. 2005.

[34] R. G. Gallager, *Low Density Parity Check Codes*. Cambridge, MA: MIT Press, 1963.

[35] A. Ghrayeb and W. E. Ryan, "Concatenated coding and iterative SOVA decoding with PR4 signaling", in *Proc. IEEE Int. Conf. Commun*., 2000, pp. 597-601.

[36] M. P. C. Fossorier, F. Burkert, S. Lin, and J. Hagenauer, "On the equivalence between SOVA and max-log-MAP decodings," *IEEE Commun. Lett.*, vol. 2, no. 5, pp. 137-139, May 1998.

[37] D. J. C. MacKay, "Good error-correcting codes based on very sparse matrices," *IEEE Trans. Inf. Theory*, vol. 45, no. 2, pp. 399–431, Mar. 1999.

[38] M. C. Davey and D. MacKay, "Low-density parity check codes over GF(q)," *IEEE. Commun. Lett.*, vol. 2, no. 6, pp. 165-167, June 1998.

[39] M. C. Davey, "Error-correction using low-density parity-check codes", Ph.D. dissertation, Univ. Cambridge, Cambridge, U.K., Dec. 1999.

[40] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 599–618, Feb. 2001.

[41] S.-Y. Chung, T. J. Richardson and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity check codes using a Gaussian approximation," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 657–670, Feb. 2001.

[42] H. Song and J. R. Cruz, "Reduced-complexity decoding of Q-ary LDPC codes for Magnetic Recording," *IEEE Trans. Magn.* vol. 39, no.2, pp. 1081-1087, Mar. 2003.

[43] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498-519, Feb. 2001.

[44] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Inf. Theory*, vol. IT-27, no. 5, pp. 533–547, Sept. 1981.

[45] G. D. Forney, Jr., "On iterative decoding and the two-way algorithm," in Proc. *Int. Symp. on Turbo Codes and Relates Topics*, Brest, France, pp. 12-25, Sept. 1997.

[46] J. Chen and M. Fossorier, "Near optimum universal belief propagation based decoding of low-density parity check codes," *IEEE Trans. Commun.*, vol. 50, pp. 406–414, Mar. 2002.

[47] J. Chen and M. Fossorier, "Density evolution for two improved BP-based decoding algorithms of LDPC codes," *IEEE Commun. Lett.*, vol. 6, no. 5, pp. 208–210, May 2002.

[48] P. Hoeher, "Optimal subblock-by-subblock detection," *IEEE Trans. Commun*., vol. 43, no. 2/3/4, pp. 714-717, Feb./Mar./Apr. 1995.

[49] M. K. Cheng, J. Campello, and P. H. Siegel, "Soft-decision Reed-Solomon decoding on partial response channels," in *Proc. IEEE Global Telecommun. Conf*., 2002, pp. 1026-1030.

[50] X.-Y. Hu, E. Eleftheriou and D. M. Arnold, "Regular and irregular progressive edge-growth Tanner graphs," *IEEE Trans. Inf. Theory*, vol. 51, no. 1, pp. 386-398, Jan. 2005.

[51] J. L. Fan, *Constrained Coding and Soft Iterative Decoding*. Boston/Dordrecht/London: Kluwer Academic Publishers, 2001.

[52] Y. Nana, E. Sharon, and S. Litsyn, "Improved decoding of LDPC coded modulations," *IEEE Commun. Lett.*, vol. 10, no. 5, pp. 375-377, May 2006.

[53] W. Chang and J. R. Cruz, "Optimal channel detection for nonbinary coded partial response channels," *IEEE Trans. Commun*., vol. 57, no. 7, pp. 1892-1895, July 2009.

[54] W. Chang and J. R. Cruz, "Nonbinary LDPC codes for 4-kB sectors," *IEEE Trans. Magn.,* vol. 44, no. 11, pp. 3781-3784, Nov. 2008.

[55] Y. Li, B. Vucetic, and Y. Sato, "Optimum soft-output detection for channels with intersymbol interference," *IEEE Trans. Inf. Theory*, vol. 41, no. 3, pp. 704-713, May 1995.

[56] A. Kavcic and J. M. F. Moura, "Correlation-sensitive adaptive sequence detection," *IEEE Trans. Magn.*, vol. 34, no. 3, pp. 763–771, May 1998.

[57] J. Moon and J. Park, "Pattern-dependent noise prediction in signal-dependent noise," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 730–743, Apr. 2001.

[58] B. Vasic, K. Pedagani and M. Ivkovic, "High-rate girth-eight low-density parity-check codes on rectangular integer lattices," *IEEE Trans. Commun*., vol. 52, no. 8, pp. 1248-1252, Aug. 2004.

[59] E. Cavus, C. L. Haymes, B. Daneshrad, "An IS simulation technique for very low BER performance evaluation of LDPC codes," in *Proc. IEEE Int. Conf. Commun.*, 2006, pp. 1095-1100.

[60] M. Ivkovic, S. K. Chilappagari and B. Vasic, "Eliminating trapping sets in low-density parity-check codes by using Tanner graph covers," preprint, available online at http://arxiv.org/, May 2008.

[61] R. M. Tanner, D. Sridhara and T. Fuja, "A class of group-structured LDPC codes," in *Proc. 6th Int. Symp. Commun. Theory and Applications*, Ambleside, England, 2001, pp. 365–370.

[62] M. P. C. Fossorier, "Quasi-cyclic low-density parity-check codes from circulant permutation matrices," *IEEE Trans. Inf. Theory*, vol. 50, no. 8, pp. 1788-1793, Aug. 2004.

[63] H. Fujita and K. Sakaniwa, "Some classes of quasi-cyclic LDPC codes: properties and efficient encoding method," *IEICE Trans. Fundamentals*, vol. E88-A, no. 12, pp. 3627-3635, Dec. 2005.

[64] S. Myung, K. Yang and J. Kim, "Quasi-cyclic LDPC codes for fast encoding," *IEEE Trans. Inform. Theory*, vol. 51, no. 8, pp. 2894-2901, Aug. 2005.

[65] W. Tan, "Design of inner LDPC codes for magnetic recording channels," *IEEE Trans. Magn.,* vol. 44, no. 1, pp. 217-222, Jan. 2008.

[66] B. Vasic and O. Milenkovic, "Combinatorial constructions of low-density parity-check codes for iterative decoding," *IEEE Trans. Inf. Theory*, vol. 50, no. 6, pp. 1156-1176, June 2004.

[67] E. M. Kurtas, A. V. Kuznetsov, and I. Djurdjevic, "System perspectives for the application of structured LDPC codes to data storage devices," *IEEE Trans. Magn.,* vol. 42, no. 2, pp. 200-207, Feb. 2006.

[68] A. V. Kuznetsov and R. Venkataramani, "Macroscopic and microscopic approaches in sector failure rate estimation," *IEEE Trans. Magn.,* vol. 44, no. 1, pp. 187-192, Jan. 2008.

[69] E. M. Kurtas, J. Park, X. Yang, W. Radich and A. Kavcic, "Detection methods for data-dependent noise in storage channels", in *Coding and Signal Processing for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas, Eds. Boca Raton: CRC Press, LLC, 2005, pp. 33-1-33-17.

[70] J. Chen, A. Dholakia, E. Eleftheriou, M. P. C. Fossorier, and X. Hu, "Reduced-complexity decoding of LDPC codes," *IEEE Trans. Commun.*, vol. 53, no. 8, pp. 1288-1299, Aug. 2005.

[71] Z. A. Keirn, V. Y. Krachkovsky, E. F. Haratsch, and H. Burger, "Use of redundant bits for magnetic recording: Single-parity codes and Reed- Solomon error-correcting code," *IEEE Trans. Magn.*, vol. 40, no. 1, pp. 225–230, Jan. 2004.

[72] P. Chaichanavong and G. Burd, "On the concatenation of soft inner code with Reed-Solomon code for perpendicular magnetic recording," *IEEE Trans. Magn.*, vol. 43, no. 2, pp. 744–749, Feb. 2007.

[73] W. Tan and J. R. Cruz, "Signal processing for perpendicular recording channels with intertrack interference," *IEEE Trans. Magn.,* vol. 41, no. 2, pp. 730-735, Feb. 2005.

[74] S. Nabavi and B. V. K. Vijaya Kumar, "Two-dimensional generalized partial response equalizer for bit-patterned media," in *Proc. IEEE Int. Conf. Commun.*, Glasgow, Scotland, 2007, pp. 6249–6254.

[75] P. W. Nutter, I. T. Ntokas, B. K. Middleton and D. T. Wilton, "Effect of island distribution on error rate performance in patterned media," *IEEE Trans. Magn.,* vol. 41, no. 10, pp. 3214-3216, Oct. 2005.

[76] P. W. Nutter, I. T. Ntokas and B. K. Middleton, "An investigation of the effect of media characteristics on read channel performance for patterned media storage," *IEEE Trans. Magn.,* vol. 41, no. 11, pp. 4327-4334, Nov. 2005.

[77] M. Keskinoz, "Two-dimensional equalization/detection for patterned media storage," *IEEE Trans. Magn.,* vol. 44, no. 4, pp. 533-539, Apr. 2008.

[78] R. Ahlswede, B. Balkenhol and Ning Cai, “Parallel error correcting codes,” *IEEE Trans. Inf. Theory*, vol. 48, no. 4, pp. 959-962, Apr. 2002.

[79] H.Yagi, T.Matsushima, and S.Hirasawa, "A generalization of the parallel error correcting codes,” *in Proc. IEEE Inf. Theory Workshop*, 2006, pp.229-233.

[80] H. Kamabe, “Parallel error correcting for multi-track recording channels,” *in Digest of 11th Joint MMM-Intermag Conf.*, Jan. 2010.

Appendix A – List of Acronyms and Abbreviations

2D	Two-Dimensional
3D	Three-Dimensional
ABS	Air-Bearing Surface
APP	*a posteriori* Probability
AWGN	Additive White Gaussian Noise
BCJR	Bahl-Cocke-Jelinek-Raviv
BER	Bit-Error Rate
BP	Belief-Propagation
BPM	Bit-Patterned Media
BPMR	Bit-Patterned Magnetic Recording
CMBP	Coded Modulation Belief-Propagation
DMC	Discrete Memoryless Channel
ECC	Error Correcting Code
EMSE	Effective Mean-Squared Error
FFT	Fast Fourier Transform
FIR	Finite Impulse Response
GMR	Giant Magnetoresistive
GPR	Generalized Partial-Response
HAMR	Heat-Assisted Magnetic Recording
i.u.d.	independent uniformly distributed

IBP	Improved Belief-Propagation
ISI	Intersymbol Interference
LDPC	Low-Density Parity-Check
LLR	Log Likelihood Ratio
LMR	Longitudinal Magnetic Recording
MAP	Maximal *a posteriori*
MIR	Matched Information Rate
ML	Maximum Likelihood
MLSD	Maximum Likelihood Sequence Detector
MMS	Multilevel Modulated Signal
MMSE	Minimum Mean-Squared Error
MPEG	Modified Progressive Edge-Growth
MR	Magnetoresistive
MRC	Magnetic Recording Channel
MS	Min-Sum
MSE	Mean-Squared Error
MTR	Maximum Transition Run
NRZ	Non-Return-to-Zero
NRZI	Non-Return-to-Zero-Inverted
OBBD	Optimal Subblock-by-Subblock Detector
PDNP	Pattern-Dependent Noise Predictive
PEG	Progressive Edge-Growth

PMF	Probability Mass Function
PMR	Perpendicular Magnetic Recording
PMRC	Perpendicular Magnetic Recording Channel
PR	Partial-Response
PRML	Partial-Response Maximum Likelihood
RLL	Run Length Limited
RS	Reed-Solomon
SER	Sector-Error Rate
SISO	Soft-Input Soft-Output
SNR	Signal-to-Noise Ratio
SOVA	Soft Output Viterbi Algorithm
SUL	Soft Under Layer
TDMR	Two-Dimensional Magnetic Recording
VA	Viterbi Algorithm