Timing Recovery Based on Per-Survivor Processing

A Thesis
Presented to
The Academic Faculty

by

Piya Kovintavewat

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
October 2004

Timing Recovery
Based on Per-Survivor Processing

Approved by:

Dr. John R. Barry, Advisor

Dr. Mark A. Clements

Dr. Steven W. McLaughlin

Dr. Thomas D. Morley

Dr. Gordon L. Stüber

Date Approved: 13 October 2004

To Punnee, Kiat,

and others in my beloved family.

ACKNOWLEDGEMENTS

First of all, I would like to express my deepest gratitude to my advisor, Dr. John R. Barry, for granting me with the wonderful opportunity to do this interesting research work, for triggering many ideas that this thesis is built upon, for providing invaluable expertise, and for supporting me during my stay at GaTech. He always makes me feel welcome to discuss both research and other issues. His clear explanations on technical matters have proven to be of great assistance in my research work and made this thesis complete.

Next, I would like to thank Dr. Steven W. McLaughlin, Dr. Gordon L. Stüber, Dr. Mark A. Clements, and Dr. Thomas D. Morley for serving on my defense committee. I am also grateful for the help of the staff at the School of Electrical and Computer Engineering and at GCATT, especially Dr. David R. Hertling, Marilou Marilou, Cordai Farrar, and Suzzette E. Willingham.

I would also like to thank Seagate Technology, Pittsburgh, PA, USA, for giving me work experiences in the summers of 2001, 2002, and 2004. Many discussions, especially with Dr. M. Fatih Erden, Dr. Inci Ozgunes, Dr. Erozan M. Kurtas, Dr. Jongseung Park, Dr. Alexander V. Kuznetsov, Dr. Walter R. Eppler, and Dr. Xueshi Yang have been invaluable and stimulating.

My family has always given me strong support and encouragement in my studies and in other aspects of my life. I would like to thank everyone in my family, especially my parents (Punnee and Kiat). They have always taught me to be patient, diligent, concentrated, and cheerful. I would also like to thank the Thai government for its financial support during my Ph.D. study.

Special thanks are also given to Aravind, Badri, Pornchai, Joon, Renato, and

others in the Communications and Information Theory Laboratory at GaTech. These people had engaged me in many discussions and helped bring understanding to many research questions. Finally, I would like to thank all my friends for supporting and encouraging me throughout my stay at GaTech.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

At some point in a digital communications receiver, the received analog signal must be sampled. Sampling at the wrong times can have a devastating impact on performance. The process of synchronizing the sampler with the received analog signal is known as timing recovery. Conventional timing recovery techniques are based on a decision-directed phase-locked loop (PLL). They are adequate only when the operating signal-to-noise ratio (SNR) is sufficiently high, but recent advances in error-control coding have made it possible to communicate reliably at very low SNR, where conventional techniques fail. This thesis develops new techniques for timing recovery that are capable of working at low SNR.

We propose a new timing recovery scheme based on per-survivor processing (PSP), which jointly performs timing recovery and equalization, by embedding a separate decision-directed PLL into each survivor of a Viterbi algorithm. The proposed scheme is shown to perform better than conventional timing recovery, especially when the SNR is low and the timing error is large. An important advantage of this technique is its amenability to real-time implementation.

We also propose a new iterative timing recovery scheme that exploits the presence of the error-control code; in doing so, it can perform even better than the PSP scheme described above, but at the expense of increased complexity and the requirement of batch processing. The proposed iterative timing recovery scheme is realized by embedding the timing recovery process into a trellis-based soft-output equalizer using PSP. Then, this module iteratively exchanges soft information with the error-control decoder, as in conventional turbo equalization. The resulting system jointly performs the functions of timing recovery, equalization, and decoding. The proposed iterative

timing recovery scheme is shown to perform better than previously reported iterative timing recovery schemes, especially when the timing error is severe.

Performance analysis of iterative timing recovery schemes is difficult because of their high complexity. We propose to use the extrinsic information transfer (EXIT) chart as a tool to predict and compare their performances, considering that the bit-error rate computation takes a significant amount of simulation time. Experimental results indicate that the system performance predicted by the EXIT chart coincides with that obtained by simulating data transmission over a complete iterative receiver, especially when the coded block length is large.

Finally, we investigate the performance of the proposed timing recovery schemes in a magnetic recording system. This system is considered because magnetic recording is a primary method of storage for a variety of applications, including desktop, mobile, and server systems. This experiment will help us decide whether or not the proposed schemes are worth being employed in real-life applications, if compared to the conventional schemes used in today's magnetic recording read-channel chip architectures. Simulation results show that the proposed schemes perform better and achieve higher storage capacity than conventional schemes.

CHAPTER 1

INTRODUCTION

At some point in a digital communications receiver, the received analog signal must be sampled before performing equalization and decoding. Sampling at the wrong times can have a devastating impact on overall performance. The process of synchronizing the sampler with the received analog signal is known as *timing recovery*. Note that, in this work, we focus only on the problem of synchronization in baseband transmission systems. As a consequence, we do not consider carrier recovery in this work.

To facilitate timing recovery, a clock signal may be sent separately from the data signal, thus increasing the bandwidth, transmitted power, and so forth. To avoid these inefficiencies, it is customary to recover the clock from the data signal itself, which is referred to as *self-timing* [9]. This method relies primarily on the timing information present in the received data signal. As the received data signal practically undergoes many kinds of impairments, such as timing offset[1], frequency offset[2], additive noise, etc., the amount of timing information embedded in the received data signal tends to be decreased. Furthermore, since most timing recovery schemes are based on a decision-directed phase-lock loop (PLL) [9], the performance of timing recovery is then a strong function of the reliability of decisions, and hence, of the operating signal-to-noise ratio (SNR). This implies that timing recovery at low SNR is even more difficult. Thus, the need for *efficient* timing recovery schemes becomes increasingly important because improving the performance of timing recovery will

[1]Timing offset happens when the actual and the expected arrival times of the k-th pulse do not coincide.

[2]Frequency offset occurs when the transmitted and the receiver clock frequencies differ from each other by a fraction of a percent.

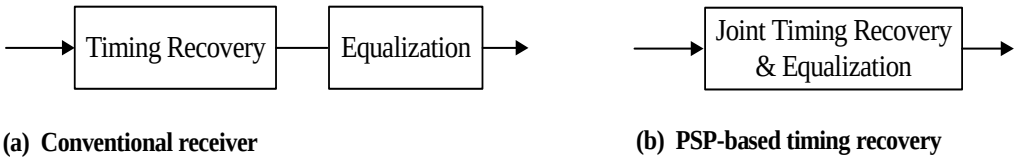


Figure 1: Front-end receiver structures for uncoded systems: (a) conventional receiver and (b) PSP-based timing recovery.

give rise to improved reliability of an entire system. Therefore, in this work, we focus on developing new timing recovery schemes that perform better than conventional schemes for the systems with and without error-correction codes (ECCs).

1.1 Uncoded Systems

For systems without ECCs (i.e., *uncoded* systems), the receiver performs two main tasks, namely, timing recovery and equalization. The former is usually performed by *conventional timing recovery*, which takes the form of a PLL, whereas the latter is practically performed by a Viterbi detector [24].

Theoretically, joint maximum-likelihood (ML) estimation of the timing offset and the data sequence is a preferred method of synchronization [48] but its complexity is huge. Georghiades and Snyder [27] applied the expectation-maximization (EM) algorithm to jointly estimate the timing offset and the data sequence in the case of a constant timing offset with an ideal, uncoded system without intersymbol interference (ISI). However, this method is still complicated. In practice, a conventional receiver performs timing recovery and ML equalization separately, as depicted in Figure 1(a). Specifically, conventional timing recovery is based on a PLL that relies on the decision provided by its own symbol detector, which can be either a Viterbi detector with a *short* decision delay or a memoryless multi-level slicer. Nevertheless, the Viterbi detector has a fundamental trade-off between reliability and the decision delay, whereas the memoryless multi-level slicer might yield an unreliable decision.

To improve the performance of conventional timing recovery, a reliable decision with *zero* decision delay can be extracted by utilizing the already-given information inside the trellis structure [24]. Specifically, each state transition in the trellis uniquely specifies a corresponding symbol. Thus, at least one state transition in each trellis stage will correspond to a correct decision. Utilizing that decision for the timing update operation will improve the performance of timing recovery. The idea of using the information available in the trellis to estimate other unknown parameters is known as *per-survivor processing* (PSP) [65].

PSP is a technique of jointly estimating a data sequence and unknown parameters, such as channel coefficients, the carrier phase, and so on. It was first used in the application of reduced-state sequence estimation [22]. The general PSP concept and its various applications were later introduced by Raheli, Polydoros, and Tzou [65]. PSP has been employed in many applications, including channel identification, adaptive ML sequence detection, and phase/carrier recovery [4, 13, 18, 21, 41, 65, 66, 83]. In addition, Iltis [31] utilized the PSP concept in conjunction with the extended Kalman filter to jointly estimate the timing offset and the data sequence in the case of a constant timing offset with the ISI channel, but this method is too complex.

In this work, we propose a new timing recovery scheme based on PSP, denoted as *PSP-based timing recovery* [36], for uncoded partial response (PR) [81] channels. The proposed scheme jointly performs timing recovery and ML equalization, as shown in Figure 1(b). Unlike the method proposed in [31], our scheme has lower complexity and can also deal with time-varying timing offsets. Simulation results indicate that for low to moderate SNRs, PSP-based timing recovery performs better than conventional timing recovery, especially when the timing jitter is severe or when operating in a system that requires fast convergence.

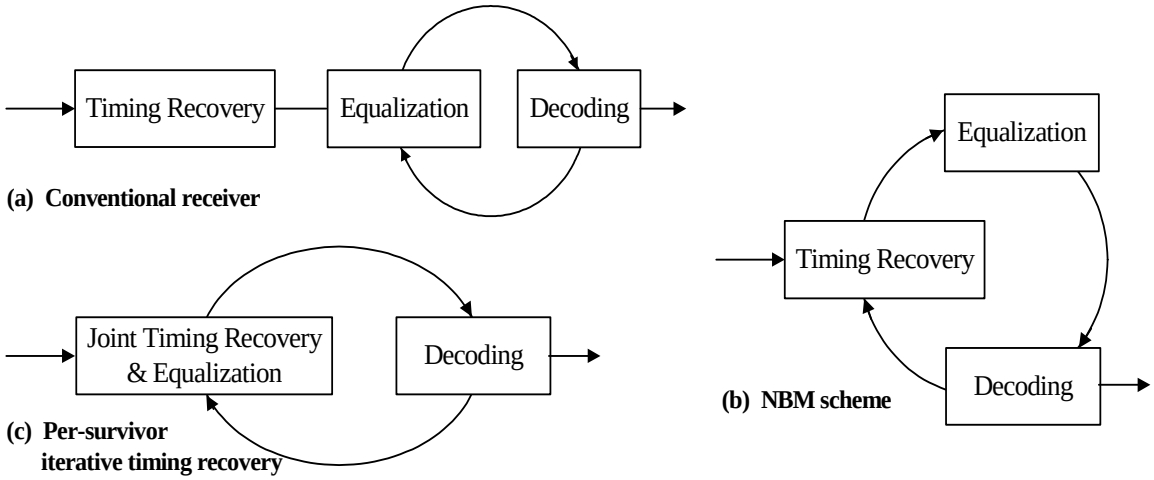


Figure 2: Front-end receiver structures for coded systems: (a) conventional receiver, (b) NBM scheme, and (c) per-survivor iterative timing recovery.

1.2 Coded Systems

Iterative ECCs, such as turbo codes [10] and low-density parity-check (LDPC) codes [26], allow reliable operation at low SNR because of their large coding gains [10, 30, 86]. Furthermore, the principle of iterative decoding can also be extended to include equalization, which is commonly known as *turbo equalization* [67, 79]. This means that timing recovery must also function at SNR lower than ever before. Note that lower SNR is a desirable property because it helps reduce the cost of operation, and in magnetic recording systems for example, allows for higher storage capacity.

Generally, the decisions from the ECC decoder are more reliable than those from the PLL, but the decoding process usually introduces a large decision delay. At high SNR, we can afford to use the decisions from the PLL in order to avoid a large decision delay. Nonetheless, at low SNR, we need to exploit the presence of ECCs so as to have reliable decisions for timing recovery. Although this will be more complex and introduce a large decision delay, it should provide better performance. That is why a conventional receiver, which performs timing recovery and error-correction decoding separately, as depicted in Figure 2(a), fails to work properly at low SNR.

Theoretically, joint ML estimation of the timing offset and the data sequence, which jointly performs timing recovery, equalization, and error-correction decoding, is a preferred method of synchronization [48], but its complexity is problematic. A solution based on the EM algorithm [27, 57] is also complex. Fortunately, a solution to this problem with complexity comparable to the conventional receiver has been proposed by Nayak, Barry, and McLaughlin [56], which will be referred to as the *NBM scheme*, as shown in Figure 2(b). The NBM scheme is realized by embedding the timing recovery step inside the turbo equalizer [67, 75] so as to perform timing recovery, equalization, and error-correction decoding jointly. The key idea of the NBM scheme is as follows. At each turbo iteration, the turbo equalizer will produce the decisions that are more reliable than the decisions from the PLL. These better decisions are fed back to the timing recovery unit to improve the timing estimates. Then, the new timing estimates are used to refine the samples. These better samples will be employed to improve the performance of the turbo equalizer in the next iteration. This process repeats as many iterations as needed. In summary, the turbo equalizer benefits from better samples, and timing recovery benefits from better decisions.

At high SNR, the decisions provided by a symbol detector used in a PLL are reliable enough for the timing recovery unit to perform well. Thus, the conventional receiver is sufficient to be used in a system operating at high SNR because of its simplicity. On the other hand, at low SNR and moderate to severe timing offset models, timing recovery is very difficult because of the phenomenon called a *cycle slip* [9]. When a cycle slip occurs, the receiver adds or drops symbols, and this causes a burst of errors in data detection process. In the presence of a cycle slip, the ECC decoder usually fails to decode. This explains why the conventional receiver does not perform well at low SNR. However, the NBM scheme has the ability to correct a cycle slip [7, 55, 56]. In fact, when a cycle slip occurs, the timing recovery unit does not suddenly add or drop symbols. Instead, it gradually loses track of the actual

timing offset until it settles down at the offset corresponding to multiples of symbol durations. Therefore, as iterations increase, the area of the boundary zone between the actual timing offset and the multiple symbol offset decreases, and this boundary moves towards the end of the data packet [55]. In other words, the portion of the data packet affected by a cycle slip gradually reduces and, eventually, disappears. To reduce the number of iterations needed to correct a cycle slip, some simple cycle slip detection and correction algorithms have been proposed in [55, 56]. Although the NBM scheme outperforms the conventional receiver [7, 55, 56], it requires a large number of iterations to provide a good performance even with a cycle slip detection and correction algorithm as used in [56], especially when the timing error is severe.

To improve the performance of the NBM scheme, we propose a new iterative timing recovery scheme based on PSP, which will be referred to as *per-survivor iterative timing recovery* [35], for coded PR channels. The proposed scheme will jointly perform timing recovery, equalization, and error-correction decoding, as illustrated in Figure 2(c). It is realized by first developing a per-survivor Bahl, Cocke, Jelinek, and Raviv (BCJR) [5] equalizer, denoted as "PSP-BCJR," by embedding the timing recovery step inside the BCJR equalizer based on PSP. Then, per-survivor iterative timing recovery iteratively exchanges soft information between PSP-BCJR and an error-correction decoder. It will be shown in simulation that per-survivor iterative timing recovery provides a significant performance improvement over other iterative timing recovery schemes [35], especially when the timing jitter is large. This is because per-survivor iterative timing recovery can automatically correct a cycle slip much more efficiently than other schemes.

Since performance analysis of iterative timing recovery [7] schemes is difficult because of their complexity, a time-consuming simulation in terms of the bit-error rate (BER) is usually a solution to compare their performances. The *extrinsic information transfer chart* (EXIT chart) was proposed by ten Brink [80] as a tool for predicting

the convergence behavior of turbo codes. In this work, we propose to use the EXIT chart as a tool to predict and compare the performance of iterative timing recovery schemes [34] since the BER computation takes a considerable amount of simulation time. Simulation results show that the system performance predicted by the EXIT chart coincides with that obtained by simulating data transmission over a complete iterative receiver, especially when the coded block length is large.

Because timing recovery is closely related to carrier recovery, there are many recent works that utilize an iterative concept to estimate a constant carrier phase [1, 2, 3, 11, 14, 43, 44, 49, 58, 60, 76, 84, 88]. However, no previous work applied the PSP concept in iterative detection to solve the problem of timing recovery.

1.3 Organization

The rest of this thesis is organized as follows. Chapter 2 briefly describes how conventional timing recovery, which is based on a PLL, works. The method of designing the PLL gain parameters based on a linearized model of PLL is also given. It will be shown in simulation that for low to moderate SNRs, conventional timing recovery does not perform well, especially when the timing error is large or when operating in a system that requires fast convergence.

The PSP-based timing recovery scheme is proposed in Chapter 3 for uncoded PR channels so as to improve the performance of conventional timing recovery without exploiting the presence of ECCs. Its architecture is fully explored and its performance is compared with conventional timing recovery. The convergence behavior of different timing recovery schemes is studied, which indicates that PSP-based timing recovery can achieve faster convergence than conventional timing recovery. Then, a reduced-complexity version of PSP-based timing recovery will be given and investigated.

Chapter 4 deals with the problem of timing recovery operating at low SNR. In this case, we consider coded PR channels. It will be shown through simulation that

the conventional receiver, which performs timing recovery and error-correction decoding separately, does not perform well at low SNR. To solve this problem, we propose per-survivor iterative timing recovery for coded PR channels. The performance comparison among different iterative timing recovery schemes is provided. A reduced-complexity version of per-survivor iterative timing recovery is also given and investigated. Since performance analysis of iterative timing recovery schemes is difficult because of their complexity, we explain how to use the EXIT chart analysis instead of BER to predict and compare their performances.

Chapter 5 is devoted to the application of magnetic recording systems. This application is considered because magnetic recording is a primary method of storage for a variety of applications, including desktop, mobile, and server systems. Timing recovery in magnetic recording systems is an increasingly critical problem because of the growing data rate to be supported. Improving the performance of timing recovery gives rise to improved reliability of an entire recording system, which in turn results in an increased storage capacity. Hence, this experiment will help us decide whether or not the proposed timing recovery schemes are worth being employed in real-life applications, if compared to the conventional schemes used in today's magnetic recording read-channel chip architectures. This chapter begins with briefly reviewing the background of magnetic recording systems. A *realistic* magnetic recording channel model, which represents all the components that are employed in magnetic recording channels, is introduced. The method of designing the *target*[3] [9, 53] and its corresponding equalizer is also given. Then, the proposed timing recovery schemes will be investigated and compared with conventional schemes, based on the realistic magnetic recording channel model with and without ECCs. Finally, Chapter 6 contains the conclusion of the thesis and possibilities for further research.

[3]A linear filter with a small number of taps that is designed to closely match the overall channel impulse response as much as possible without excessive noise enhancement.

CHAPTER 2

BASICS OF TIMING RECOVERY

This chapter briefly reviews the concept of timing recovery and explains how conventional timing recovery that is based on a PLL works. A method of designing the PLL gain parameters based on a linearized model of PLL is given. It will be shown in simulation that for low to moderate SNRs, conventional timing recovery does not perform well in uncoded systems, especially when the timing error is large or when operating in a system that requires fast convergence.

2.1 Introduction

Timing recovery is the process of synchronizing the sampler with the received analog signal. Sampling at the wrong times can have a devastating impact on overall performance. Therefore, the quality of synchronization is very important.

Most practical timing recovery schemes are based on a PLL [9], which consists of a timing error detector (TED), a loop filter, and a voltage-controlled oscillator (VCO). Typically, there are two configurations of timing recovery, namely, *deductive* and *inductive* timing recovery schemes [9], depending on whether the timing information is extracted before or after the sampler. Deductive timing recovery directly extracts the *timing tone* [8] from the incoming signal before the sampler, as shown in Figure 3, where a PLL is used to reduce the effect of timing jitter [8]. Inductive timing recovery, on the other hands, employs a feedback loop using a PLL to extract the timing information, as depicted in Figure 4. The key advantage of inductive timing recovery is that it can be implemented digitally.

In this work, only inductive timing recovery is considered, which will be referred

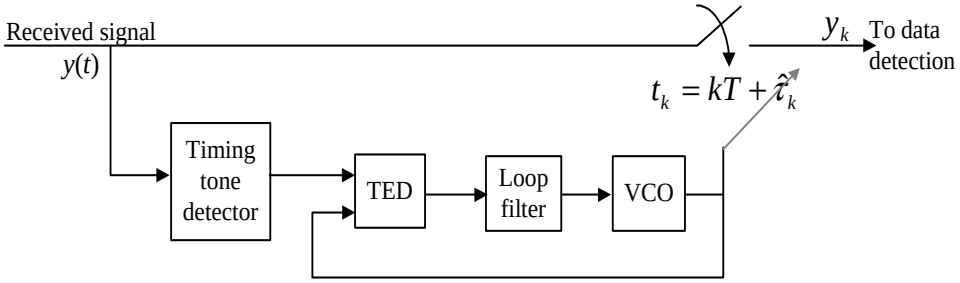


Figure 3: Deductive (or feed-forward) timing recovery.

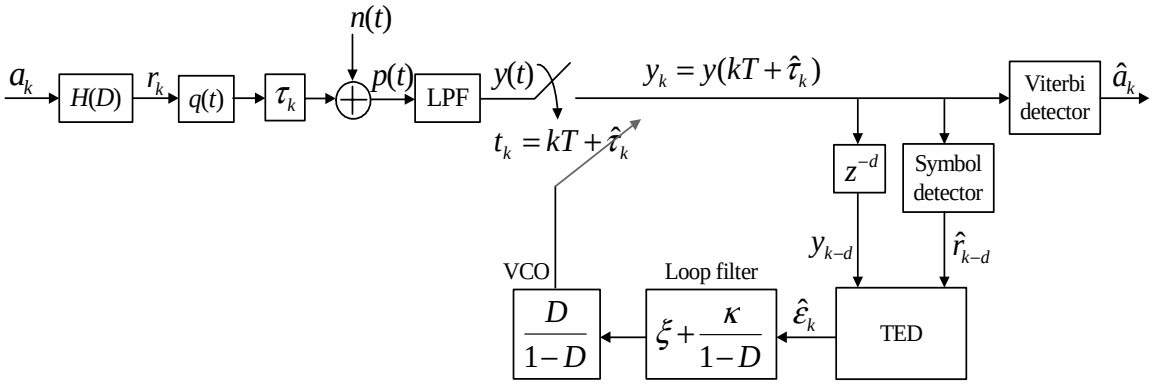


Figure 4: The perfectly equalized channel model with inductive (feedback) timing recovery.

to as *conventional timing recovery*. A reader interested in deductive (or feed-forward) timing recovery can refer to [64] for a brief discussion.

2.2 Conventional Timing Recovery

Consider the perfectly equalized channel model (also referred to as an *ideal* channel model) shown in Figure 4. An input data sequence $a_k \in \{\pm 1\}$ with bit period T is filtered by the channel represented by $H(D) = \sum_{k=0}^{\nu} h_k D^k$, where h_k is the k-th channel coefficient, D is the delay operator, and ν is channel memory. The readback signal, $p(t)$, can then be written as

$$p(t) = \sum_k r_k q(t - kT - \tau_k) + n(t), \tag{1}$$

where $r_k = \sum_i a_{k-i} h_i$ is the noiseless channel output, $q(t) = \sin(\pi t/T)/(\pi t/T)$ is an ideal zero-excess-bandwidth Nyquist pulse [8], τ_k is the k-th unknown timing offset, and $n(t)$ is additive white Gaussian noise (AWGN) with two-sided power spectral density $N_0/2$. Unless otherwise specified, the timing offset τ_k used throughout this work is modeled as a random walk [4] according to

$$\tau_{k+1} = \tau_k + w_k, \tag{2}$$

where w_k is an independent and identically distributed (*i.i.d.*) zero-mean Gaussian random variable with variance σ_w^2, i.e., $w_k \sim \mathcal{N}(0, \sigma_w^2)$, and σ_w determines the severity of the timing jitter. The random walk model is chosen because of its simplicity and its ability to represent a variety of channels by changing only one parameter.

At the front-end receiver, the readback signal is filtered by a low-pass filter[1] (LPF), whose impulse response is $q(t)/T$ (i.e., a cutoff frequency is at $1/(2T)$), to eliminate the out-of-band noise and is then sampled at time $kT + \hat{\tau}_k$, creating

$$y_k = y(kT + \hat{\tau}_k) = \sum_i r_i q(kT + \hat{\tau}_k - iT - \tau_i) + n_k, \tag{3}$$

where $\hat{\tau}_k$ is an estimate of τ_k (or the k-th sampling phase offset), and n_k is an *i.i.d.* zero-mean Gaussian random variable with variance $\sigma_n^2 = N_0/(2T)$, i.e., $n_k \sim \mathcal{N}(0, \sigma_n^2)$.

A decision-directed TED [9] is used to compute the receiver's estimate of the timing error $\epsilon_k = \tau_k - \hat{\tau}_k$, which is the misalignment between the phase of the received signal and that of the sampling clock. Several TED algorithms have been proposed in the literature [9, 46], depending on how they incorporate the information available at the TED input. Typically, the overall performance of timing recovery is dominated by the effectiveness of the TED. In this work, we consider the well-known Mueller and Müller (M&M) TED algorithm [54], where the estimated timing error is given

[1]For a perfect bandlimited system where all of the signal energy is confined within the $|f| \leq 1/(2T)$ band, a low-pass filter also provides the sufficient statistic [52] as a matched filter [8] does.

by

$$\hat{\epsilon}_k = K_T\{y_k\hat{r}_{k-1} - y_{k-1}\hat{r}_k\}, \tag{4}$$

where $\hat{r}_k$ is an estimate of r_k. The constant K_T is used to ensure that there is no bias at high SNR so that $E[\hat{\epsilon}_k|\epsilon] = \epsilon$ (or, in other words, the slope of the S-curve [9] is unity at the origin). As shown in (4), the TED performance depends on the decisions $\{\hat{r}_k\}$. Therefore, timing recovery performance is a strong function of the reliability of decisions and hence of the operating SNR. This explains why the symbol detector used in the timing loop is a Viterbi detector with a short decision delay, dT, instead of a memoryless multi-level slicer in most real-life applications [15].

Next, the estimated timing error $\hat{\epsilon}_k$ is filtered by a loop filter to eliminate the noise in the timing error signal. Then, the next sampling phase offset is updated by a second-order PLL according to [9]

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \kappa\hat{\epsilon}_k, \tag{5}$$

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi\hat{\epsilon}_k + \hat{\theta}_{k+1}, \tag{6}$$

where $\hat{\theta}_k$ represents an estimate of *frequency error* [15], and ξ and κ are the PLL gain parameters [9], which determine the loop bandwidth and the rate of convergence. The larger the value of PLL gain parameters, the larger the loop bandwidth, the faster the convergence rate, and thus the more the noise allowed to perturb the system. Note that a first-order PLL can only handle phase error, not frequency error. A first-order PLL update equation is easily obtained by setting $\kappa = 0$ in (5).

In practice, timing recovery is performed in two modes, namely, *acquisition* and *tracking* modes. An acquisition mode is performed at the beginning of the data sector with the aid of a known data pattern called a *preamble* [15] (or a training sequence) to acquire the initial phase and frequency estimates. Hence, a tracking mode is performed using the samples corresponding to transmitted unknown data so as to refine these initial estimates. Since the preamble is known at the receiver, large

values of ξ and κ can be used to expedite the convergence rate. However, the values of ξ and κ should be lowered during tracking mode so as to reduce the effect of the noise [71]. Therefore, designers must tradeoff between the loop bandwidth and the convergence rate when designing ξ and κ.

2.3 Design of PLL Gain Parameters

From the simulation point of view, the best way to choose the PLL gain parameters (both ξ and κ) is to optimize them based on minimizing the BER at the detector output. Nevertheless, this method is impractical and time-consuming. Instead, one usually designs ξ and κ based on a linearized model of PLL [9]. One possible criterion is to choose ξ and κ so that the system response can catch a phase and/or frequency change in the system within "C" samples (or bit periods). It should be noted that this criterion can also be viewed as the rate of convergence, i.e., the smaller the C, the faster the convergence rate.

2.3.1 Linear Analysis of First-Order PLL

A first-order PLL is of restricted practical interest because it can only handle phase error, not frequency error. Nonetheless, its analysis is a good start for understanding a higher-order PLL. In this analysis, the phase error is modeled as a *step* function according to $\tau_k = T$ for $k \geq 0$, and $\tau_k = 0$ for $k < 0$.

Consider a first-order PLL update equation, which is given by

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi \hat{\epsilon}_{k-d}, \tag{7}$$

where d is a normalized delay (with respect to T) in the timing loop, $\hat{\epsilon}_k = \epsilon_k + v_k$ is the estimated timing error, $\epsilon_k = \tau_k - \hat{\tau}_k$ is the residual timing error, and v_k is the noise in the TED. By assuming that v_k is negligible, the system transfer function of (7) can be obtained by taking the Z-transform [8], i.e.,

$$G(z) = \frac{\hat{\Gamma}(z)}{\Gamma(z)} = \frac{\xi z^{-(d+1)}}{1 - z^{-1} + \xi z^{-(d+1)}}, \tag{8}$$

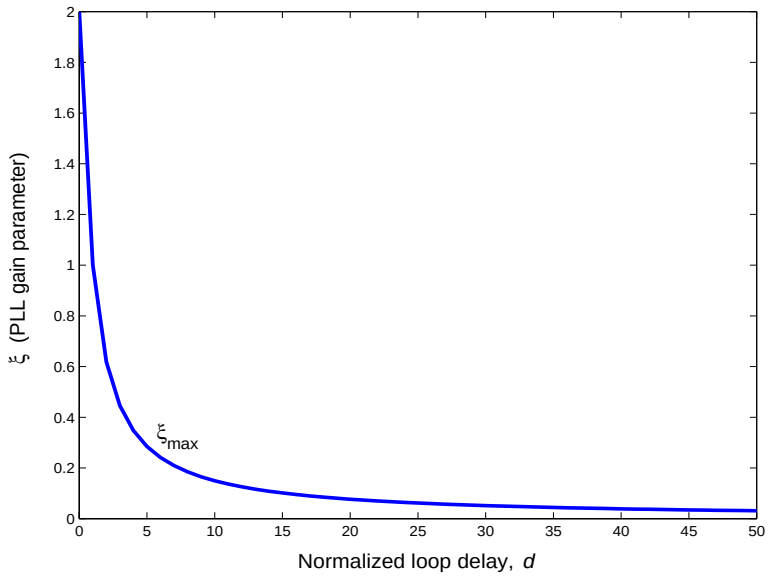


Figure 5: Maximum value of ξ satisfying the system stability for different loop delays.

where $\hat{\Gamma}(z)$ and $\Gamma(z)$ are the Z-transform of $\hat{\tau}_k$ and τ_k, respectively. Accordingly, the error transfer function (Z-transform of ϵ_k) can be written as

$$E(z) = \Gamma(z) - \hat{\Gamma}(z) = \frac{1 - z^{-1}}{1 - z^{-1} + \xi z^{-(d+1)}} \cdot \Gamma(z). \tag{9}$$

One possible criterion that may be used to choose ξ is to pick the one that satisfies both the system stability and the convergence rate, for a given d. This can be achieved by using either (8) or (9). It is done by first finding ξ's that satisfy the system stability. As seen in (8) and (9), the PLL is stable whenever all *poles* (i.e., roots of the denominator) of (8) or (9) lie inside the unit circle [8]. It can be shown that the value of ξ resulting in the system stability can be expressed as [9]

$$0 < \xi < 2 \sin\left(\frac{\pi}{4d + 2}\right). \tag{10}$$

Figure 5 shows the maximum value of ξ that retains the system stability for different delays. Apparently, the stability range of ξ's decreases dramatically as d increases. Among a set of ξ's that satisfies the system stability, we select one ξ so that the system response can catch the step response within C samples with $\pm 5\%$ tolerance.

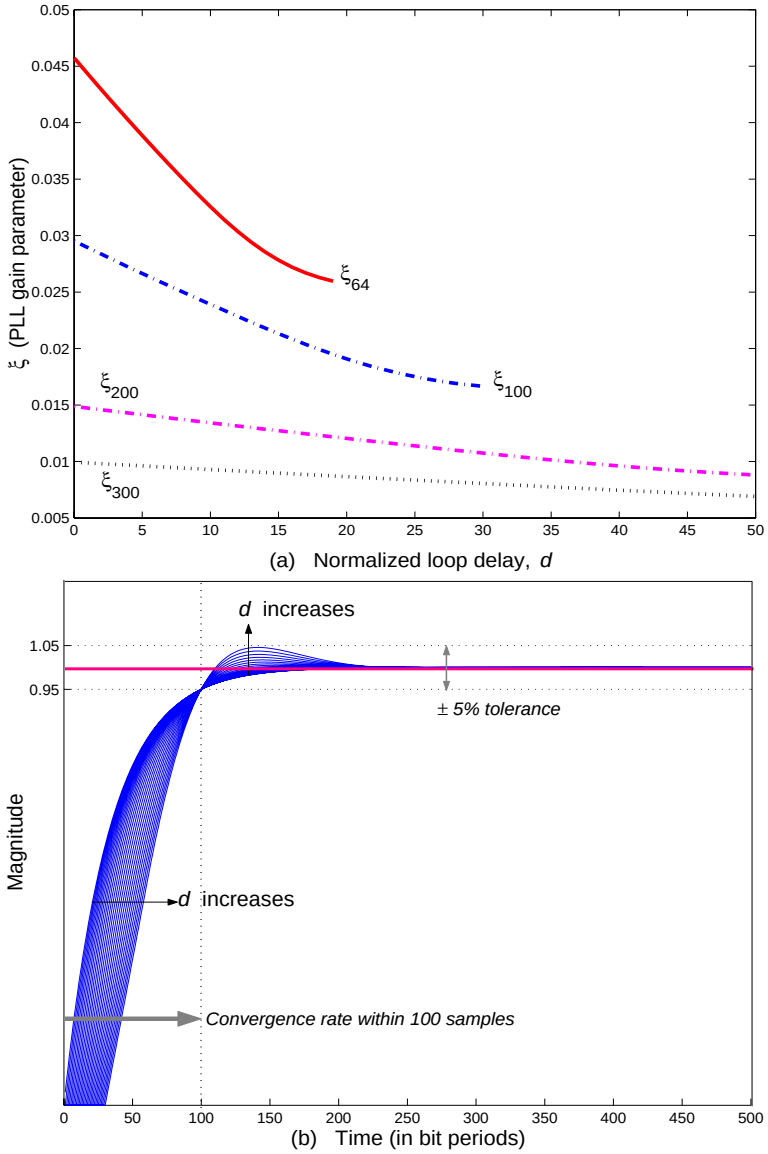


Figure 6: (a) ξ_C's satisfying the system stability and the convergence rate of C samples for different delays, and (b) the system step responses using ξ_{100} for the delays ranging from 0 to $30T$.

The 5% tolerance is introduced to relax our criterion so as to reduce the effect of the noise in the timing loop. Figure 6(a) shows ξ_C's that satisfy both the system stability and the convergence rate of C samples. Clearly, the faster the convergence rate, the larger the value of ξ. As depicted in Figure 6(a), for a given C, there are only some ξ's that satisfy the system stability up to a certain loop delay. For example, there

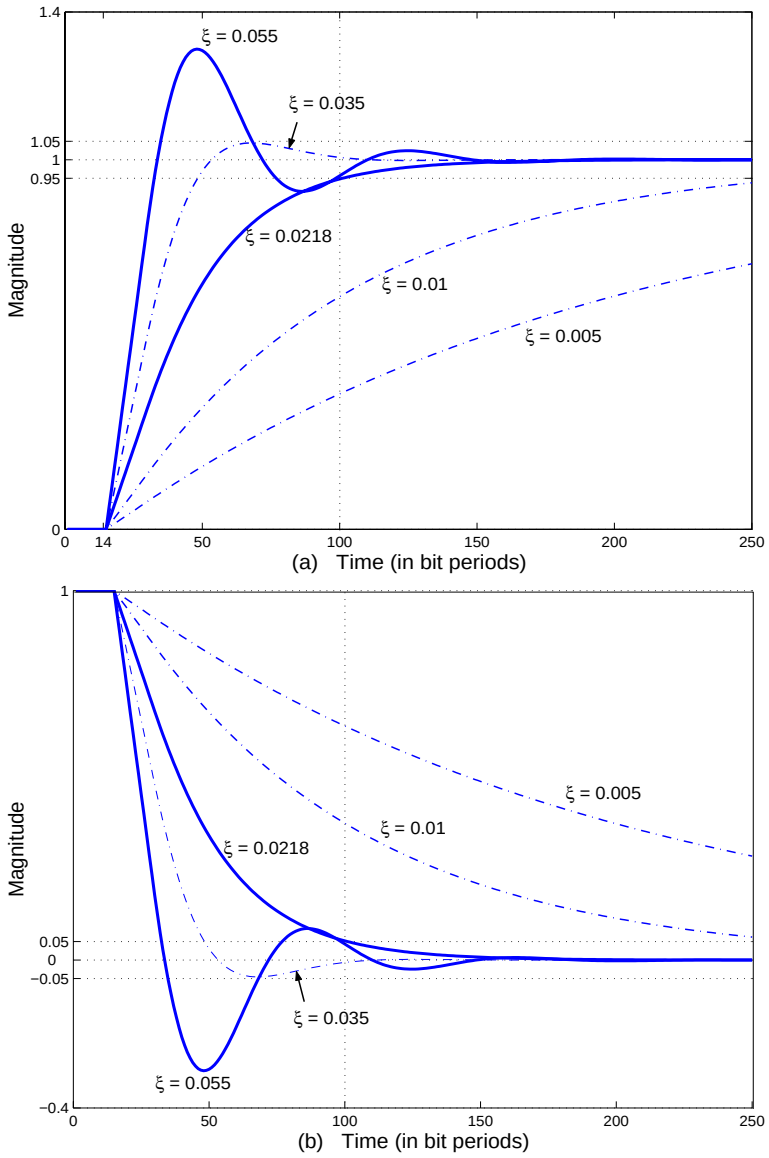


Figure 7: (a) The system step responses and (b) the error responses with different ξ's for $d = 14$.

exists ξ_{100} up to a loop delay of $30T$. Figure 6(b) shows the system step responses using (8) and ξ_{100} for $d = 0$ to 30, which coincides with our criterion.

Since we design ξ_{100} so that the system can catch the step response within 100 samples with $\pm 5\%$ tolerance, this implies that the absolute value of the magnitude of the error response $E(z)$ given in (9) should also be less than or equal to 0.05 after 100 samples, as shown in Figure 7 for $d = 14$. Observe that there are two ξ_{100}'s (i.e.,

$\xi = 0.0218$ and $\xi = 0.055$) that satisfy the convergence rate of 100 samples with $\pm 5\%$ tolerance. Nonetheless, it is desirable in practice to employ a small ξ in the timing loop to minimize the loop bandwidth, thus reducing the effect of the noise in the timing loop.

2.3.2 Linear Analysis of Second-Order PLL

When there is a frequency offset component in a system, a second-order PLL must be employed. To design the second-order PLL gain parameters (both ξ and κ), we first design ξ, for given d and C, by assuming that there is only phase error in the system. This is achieved by the method described in Section 2.3.1. After obtaining ξ, we can then design κ based on a linear analysis of second-order PLL for a given amount of frequency offset. In this analysis, the frequency error is modeled as $\tau_k = kf_d$, where f_d is the amount of frequency offset as a fraction of T.

Consider the second-order PLL update equations, which are given by

$$\hat{\theta}_{k+1} = \hat{\theta}_k + \kappa\hat{\epsilon}_{k-d}, \tag{11}$$

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi\hat{\epsilon}_{k-d} + \hat{\theta}_{k+1}. \tag{12}$$

Assuming that there is no noise in TED (i.e., using $\hat{\epsilon}_k = \epsilon_k = \tau_k - \hat{\tau}_k$), the system transfer function of a second-order PLL can be expressed as

$$G(z) = \frac{\hat{\Gamma}(z)}{\Gamma(z)} = \frac{(\xi+\kappa)z^{-(d+1)} - \xi z^{-(d+2)}}{1 - 2z^{-1} + z^{-2} + (\xi+\kappa)z^{-(d+1)} - \xi z^{-(d+2)}}, \tag{13}$$

and its corresponding error transfer function is

$$E(z) = \Gamma(z) - \hat{\Gamma}(z) = \frac{1 - 2z^{-1} + z^{-2}}{1 - 2z^{-1} + z^{-2} + (\xi+\kappa)z^{-(d+1)} - \xi z^{-(d+2)}} \cdot \Gamma(z). \tag{14}$$

Again, one possible criterion to choose κ, for given d, C, and ξ_C, is to pick the one that satisfies both the system stability and the convergence rate for a given amount of frequency offset. This can be done as follows. Given d, C, and ξ_C, we first find κ's that satisfy the system stability. As seen in (13) and (14), the PLL is stable whenever

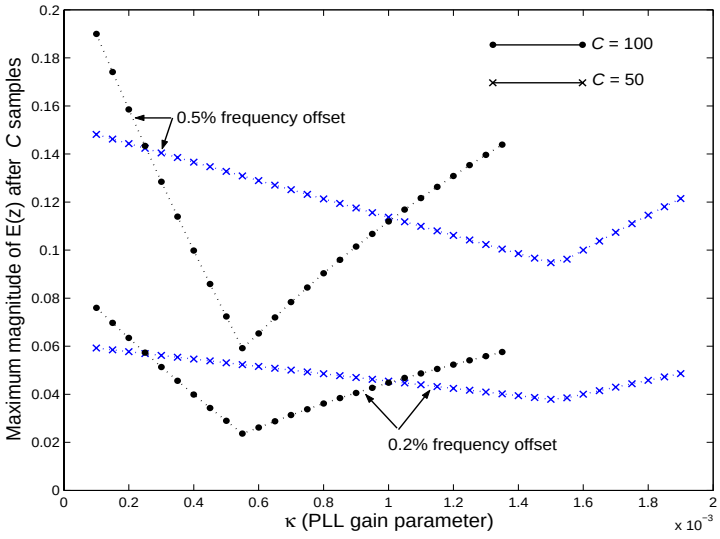


Figure 8: Maximum magnitude of $E(z)$ after C samples using $d = 14$ and ξ_C.

all poles of (13) or (14) are inside the unit circle. Among a set of κ's that satisfies the system stability, we pick one κ so that it yields the *lowest* magnitude of $E(z)$ after C samples to reduce the effect of the noise in the timing loop. Figure 8 plots the maximum magnitude of (14) after C samples as a function of κ's for $d = 14$ and ξ_C. Interestingly, it turns out that this analysis yields the same κ, regardless of the amount of frequency offset, but the corresponding magnitude of $E(z)$ is, however, proportional to the amount of frequency offset.

It should be noted that the method of designing the PLL gain parameters explained so far is based on the assumptions that the S-curve slope of (4) is unity at the origin and there is no noise in the system. Therefore, before using ξ and κ obtained from this analysis in the timing loop, we must normalize the S-curve slope of TED to be one at the origin.

2.3.3 Finding the S-curve

The S-curve or *timing function* [46] is defined as the mean of $\{\hat{\epsilon}_k\}$, assuming that all decisions are correct (i.e., $\hat{r}_k = r_k$ for all k) and the input data symbols are

uncorrelated with unit energy, i.e.,

$$S_{\mathrm{TED}}(\epsilon) = E[\hat{\epsilon}_k \,|\, \epsilon,\, \hat{r}_k = r_k \text{ for } \forall k], \tag{15}$$

where $\epsilon = \tau - \hat{\tau}$ is the timing error. Because this function looks like an "S" (rotated by 90 degrees), it is named the "S-curve." The S-curve can be used to determine the performance of TED.

For a given channel impulse response, the S-curve can be derived analytically as described in [46]. For instance, let us consider the perfectly equalized PR-IV channel model shown in Figure 4, i.e., $H(D) = 1 - D^2$. The timing function of the M&M TED for this channel can then be expressed as

$$\begin{aligned} S_{\mathrm{TED}}(\epsilon) &= E[\hat{\epsilon}_k \,|\, \epsilon,\, \hat{r}_{k-1} = r_{k-1},\, \hat{r}_k = r_k] \\ &= K_T E[r_{k-1} y_k - r_k y_{k-1}] \\ &= K_T E[(a_{k-1} - a_{k-3}) \sum_i a_i h(kT - iT - \epsilon) \\ &\qquad - (a_k - a_{k-2}) \sum_i a_i h(kT - T - iT - \epsilon)] \\ &= \frac{3T}{16} \left\{ -h(-T - \epsilon) + 2h(T - \epsilon) - h(3T - \epsilon) \right\}, \end{aligned} \tag{16}$$

where $r_k = a_k - a_{k-2}$ is the k-th noiseless channel output, $y_k = \sum_i a_i h(kT - iT - \epsilon)$ is the k-th sampler output, and $h(t) = q(t) - q(t - 2T)$ is a PR-IV pulse. The constant $K_T = 3T/16$ is introduced to ensure that the S-curve slope of (16) is unity at the origin.

On the other hand, when the channel impulse response is not known, one can still obtain the S-curve by simulation. This is done by opening the timing loop in Figure 4 (i.e., discarding a loop filter and a VCO), sampling the received signal $y(t)$ at time kT (i.e., $\hat{\tau} = 0$), and replacing τ with ϵ. Hence, we measure the time average of $\{\hat{\epsilon}_k\}$ for a given ϵ to obtain a single value of $S_{\mathrm{TED}}(\epsilon)$. We compute $S_{\mathrm{TED}}(\epsilon)$ for ϵ ranging from $-0.5T$ to $0.5T$. Eventually, the S-curve is obtained by plotting a graph between ϵ/T and $S_{\mathrm{TED}}(\epsilon)/T$.

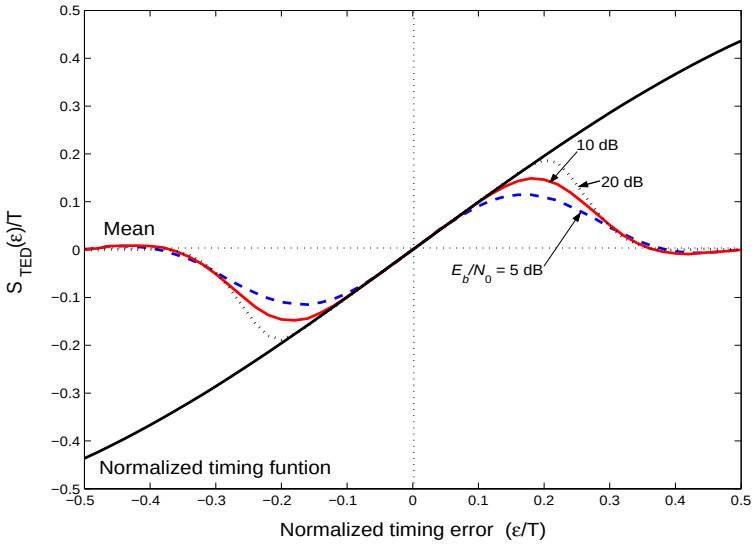


Figure 9: S-curves of the M&M TED for a PR-IV channel based on conventional timing recovery with instantaneous decision.

Figure 9 shows the S-curve of the M&M TED for a PR-IV channel based on conventional timing recovery with instantaneous *hard* decision, which is extracted by a simple ternary symbol-by-symbol decision with threshold at ± 1, i.e.,

$$\hat{r}_k = \begin{cases} 2 & \text{if } y_k > 1 \\ -2 & \text{if } y_k < -1 \\ 0 & \text{else} \end{cases} . \tag{17}$$

The curve labeled "Normalized timing function" corresponds to (16). Clearly, the timing function is odd symmetric with respect to $\epsilon = 0$. This implies that the sampling phase offset updated by a PLL using the M&M TED will settle down in the steady state at $\epsilon = 0$. It appears that simulations agree well with the timing function only for small ϵ/T's. This is because the assumption that $\hat{r}_k = r_k$ for $\forall k$ is no longer valid as ϵ/T increases. That is the reason why the range in which $S_{\text{TED}}(\epsilon)$ matches the normalized timing function at high per-bit SNR, E_b/N_0, is larger than that at low E_b/N_0.

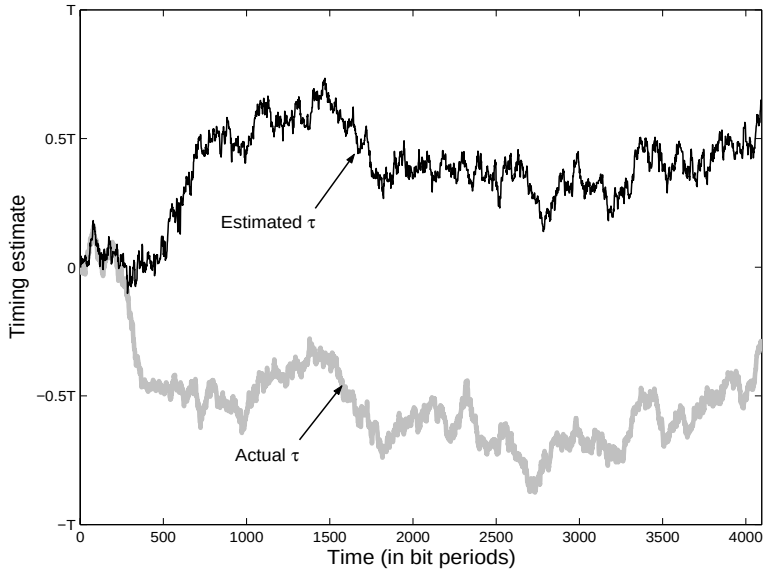


Figure 10: An example of a cycle slip.

It should be pointed out that zero crossings of the S-curve represent the equilibrium points of operation, where the PLL can track the timing offsets well. In practice, beyond $\epsilon/T = 0$, the PLL also exhibits other equilibrium points at $\epsilon = \pm T, \pm 2T, \ldots, \pm nT$, where n is an integer. The noise and other disturbances in a system can produce a large phase deviation, which may cause a transition from one equilibrium point to another. When this happens, we say that a *cycle slip* is occurred, which can cause a burst of errors in data detection process. Figure 10 demonstrates an example of a cycle slip. This figure implies that the PLL can track the *actual* timing offset, τ, well at the beginning of the data packet. Then, when a cycle slip occurs, the timing recovery unit gradually loses track of the actual timing offset until it settles down at the offset corresponding to multiples of symbol durations. In other words, a cycle slip causes the PLL to operate at another equilibrium point. That is why in this example, the estimated timing offset, $\hat{\tau}$, differs from τ by approximately T at the end of the data packet. Several approaches have been proposed to deal with a cycle slip [7, 33, 55, 56]. We will show later in this work that our proposed timing recovery schemes are also robust against a cycle slip.

2.4 Performance of Conventional Timing Recovery

Consider the perfectly equalized PR-IV channel model shown in Figure 4. Apparently, a conventional receiver performs timing recovery and ML equalization separately. Therefore, the overall performance of an uncoded system is mainly determined by how good conventional timing recovery is.

In this section, we investigate the performance of conventional timing recovery when operating in a system with and without frequency offset. We consider the case where the symbol detector used in the timing loop is a *hard slicer* (i.e., a memoryless multi-level slicer), where the instantaneous hard decision is given by (17). For data detection process, the sampler outputs $\{y_k\}$ are applied to the Viterbi detector with a decision delay of $60T$ to determine the most likely input sequence. Each BER point was computed using as many data packets as needed to collect 10000 error bits.

For the system without frequency offset, a first-order PLL is sufficient to be used for the timing update operation. In this case, we assume perfect acquisition by setting $\tau_0 = 0$ so that a preamble is not needed, and one data packet consists of 4096 data bits. Figure 11 compares the performance of the RMS timing error, $\sigma_\epsilon = \sqrt{E[(\tau_k - \hat{\tau}_k)^2]}$, and the BER as a function of E_b/N_0's. Note that the PLL gain parameter, ξ, was designed to recover the phase change within 100 symbols based on a linearized model of first-order PLL (i.e., $\xi_{100} = 0.0295$), as described in Section 2.3.1. It is evident that the larger the random walk parameter σ_w/T, the worse the performance (both in terms of σ_ϵ/T and BER). Observe that the lower the σ_ϵ/T, the lower the BER. Therefore, one can use either σ_ϵ/T or BER as a measure to compare the performance of different timing recovery schemes.

For the system with frequency offset, conventional timing recovery must employ a second-order PLL. To investigate its performance, we consider the system in a moderate system condition, e.g., with $\sigma_w/T = 0.5\%$ and 0.2% frequency offset. Again,

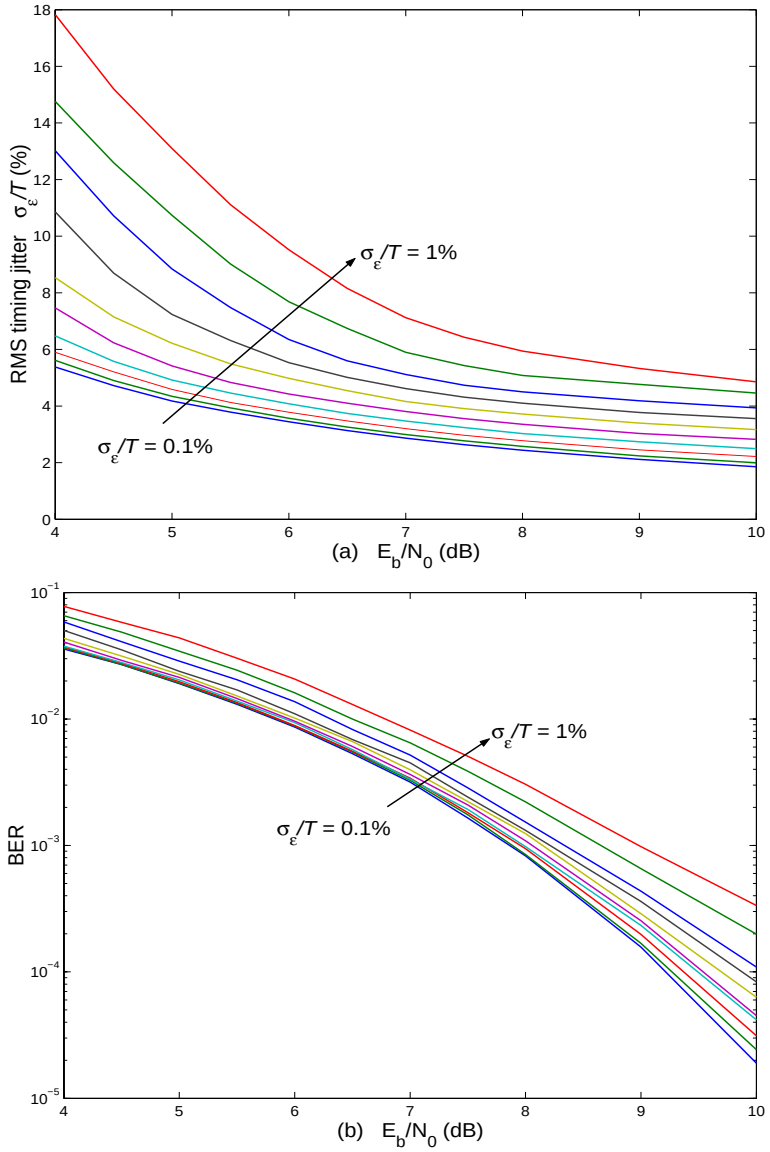


Figure 11: (a) RMS timing jitter σ_ϵ/T and (b) BER performances as a function of E_b/N_0's for the perfectly equalized PR-IV channel with different σ_w/T's (without frequency offset).

the PLL gain parameters, ξ and κ, were designed to recover phase/frequency changes within C symbols based on a linearized model of second-order PLL. The ξ's designed for $d = 0$ with $C = 50$, 100, and 256 are 0.012, 0.029, and 0.058, respectively, whereas the κ's designed for $d = 0$ with $C = 50$, 100, and 256 are 0.00015, 0.000885, and 0.00325, respectively. We also consider the case where the *same* PLL gain parameters

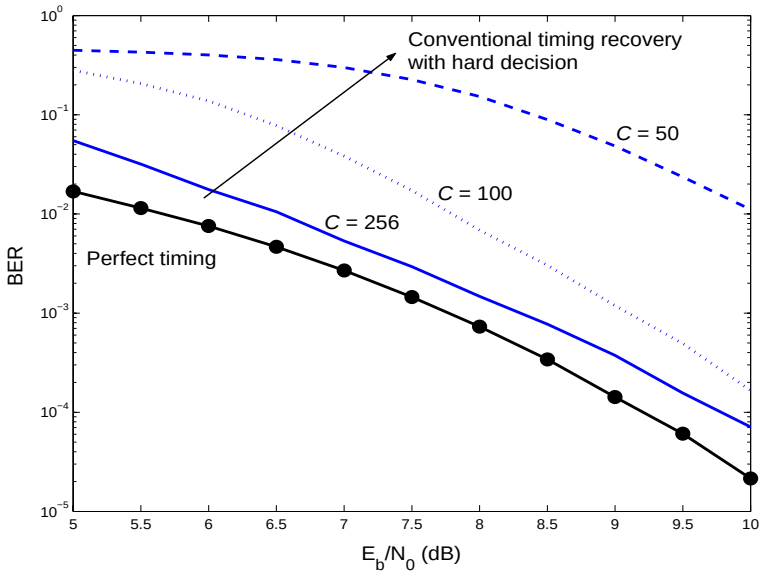


Figure 12: BER performance of conventional timing recovery for the perfectly equalized PR-IV channel with $\sigma_w/T = 0.5\%$ and 0.2% frequency offset.

are employed during both acquisition and tracking modes. One data packet consists of a C-bit preamble and 4096 data bits. Figure 12 shows the performance of conventional timing recovery using the PLL gain parameters designed for different C's. The curve labeled "Perfect timing" represents conventional timing recovery that uses $\hat{\tau}_k = \tau_k$ to sample $y(t)$. Obviously, conventional timing recovery does not work well when operating in a system that requires fast convergence, i.e., when using the PLL gain parameters designed for a small C.

As illustrated in Figures 11 and 12, it can be implied that conventional timing recovery does not perform well in uncoded systems when the timing error is large or when operating in a system that requires fast convergence. The easiest way to improve the performance of conventional timing recovery is to replace the hard slicer with either the *soft slicer* [56] or the Viterbi detector with a short decision delay [15]. However, we observed that only a small performance improvement is obtained. As a result, the need for efficient timing recovery schemes becomes increasingly crucial. The next two chapters will present new timing recovery architectures for systems with

and without error-correction codes.

2.5 Summary

We briefly explained how conventional timing recovery that is based on a PLL performs. A method of designing the PLL gain parameters based on a linearized model of PLL was given. According to our design criterion, it turns out that this method results in the same values of the PLL gain parameters, regardless of the amount of frequency offset. Finally, we have demonstrated that for low to moderate SNRs, conventional timing recovery does not perform well, especially when the timing error is large or when operating in a system that requires fast convergence.

CHAPTER 3

PER-SURVIVOR TIMING RECOVERY FOR UNCODED PARTIAL RESPONSE CHANNELS

In this chapter, a new timing recovery scheme that jointly performs timing recovery and ML equalization is proposed for uncoded PR channels. Its architecture is fully explored, and its performance is compared with conventional timing recovery. The convergence behavior of different timing recovery schemes is also studied. Finally, a reduced-complexity version of the proposed scheme is given and investigated.

3.1 Introduction

Theoretically, joint ML estimation of the timing offset and the data sequence is a preferred method of synchronization [48] but this approach is very complex. A solution based on either the expectation-maximization (EM) algorithm [27, 57] or the extended Kalman filter [31] is still complicated. In practice, a conventional receiver performs timing recovery and ML equalization separately, as shown in Figure 4. Specifically, conventional timing recovery is based on a PLL that relies on the decision provided by its own symbol detector, which can be either a Viterbi detector with a short decision delay or a memoryless multi-level slicer. However, the Viterbi detector has a fundamental trade-off between reliability and the decision delay, whereas the memoryless multi-level slicer might yield an unreliable decision.

To improve the performance of conventional timing recovery, a reliable decision with zero decision delay can be extracted by utilizing the already-given information inside the trellis structure [24]. Specifically, each state transition in the trellis uniquely specifies a corresponding symbol. Hence, at least one state transition in each trellis

stage will correspond to a correct decision. Utilizing that decision for the timing update operation will improve the performance of timing recovery. The idea of using the information available in the trellis to estimate other unknown parameters is known as *per-survivor processing* (PSP) [66]. PSP has been employed in many applications including channel identification, adaptive ML sequence detection, and phase/carrier recovery [4, 13, 18, 21, 41, 65, 66, 83].

With PSP, we propose *PSP-based timing recovery* [36] for uncoded PR channels, which jointly performs timing recovery and ML equalization, as shown in Figure 1(b). The proposed scheme will take the received analog signal as its input, perform timing recovery and ML equalization jointly, and then output the most likely input sequence.

3.2 System Description

We use the same perfectly equalized PR-IV channel model as shown in Figure 4 (i.e., with $H(D) = 1 - D^2$). We also assume perfect acquisition by setting $\tau_0 = 0$. Because our model has no frequency offset component, the sampling phase offset can then be updated by a first-order PLL according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi \frac{3T}{16} \{ y_k \hat{r}_{k-1} - y_{k-1} \hat{r}_k \}. \tag{18}$$

where $\hat{r}_k$ is an estimate of $r_k \in \{0, \pm 2\}$.

As shown in (18), the performance of conventional timing recovery depends on the quality of the decision $\hat{r}_k$ provided by a symbol detector used in the timing loop. It is evident from Figure 4 that the total delay in the timing loop results from the decision delay, dT, introduced by the symbol detector. The instantaneous hard decision (i.e., $d = 0$) can be extracted by a memoryless three-level quantization of y_k as given in (17), but it might yield an unreliable decision. An improved decision can be obtained from the Viterbi detector with a short decision delay of dT. This is done by choosing the *best* survivor path at each time instant, and then the tentative decision, $\hat{r}_{k-d}$, is found by moving d steps backward along that survivor path. Apparently, there is a

trade-off between reliability and the decision delay, since reliability can be improved by increasing the decision delay. However, a large delay is undesirable because it slows the PLL's response to the time-varying timing offsets.

Another method to obtain a good decision with zero decision delay is to utilize the PSP technique, which will be discussed in the next section.

3.3 PSP-Based Timing Recovery

PSP-based timing recovery works in a similar fashion as the Viterbi algorithm does, except with an additional timing update operation. The key idea of PSP-based timing recovery is to sample the received analog signal $y(t)$ using different sampling phase offsets associated with each state transition. Thus, the branch metrics at each stage of the trellis are calculated based on the sampling phase offset of the starting state. Additionally, each survivor of the Viterbi algorithm maintains its own estimate of the timing offset, and this estimate is updated according to the history data associated with the survivor path. For simplicity, we first restrict ourselves to an M&M TED algorithm when performing the timing update operation. As a result, we will refer to PSP-based timing recovery with an M&M TED as "PSP-MM."

3.3.1 PSP-MM Algorithm

Figure 13 shows the PSP-MM algorithm, where the lines beginning with * are the additional steps beyond the conventional Viterbi algorithm. Note that the constant $3T/16$ in (A-10) is only for the PR-IV channel, and it can also be included in the PLL gain parameters. The details on how PSP-MM performs can be explained as follows.

Consider the PR-IV trellis structure in Figure 14. Let $\Psi_k = \{a_{k-1}\, a_{k-2}\}$ denote the *state* at time k (or the k-th *stage*). There are $Q = 2^\nu = 4$ states in this trellis labeled as state 0 to state 3, where ν is the PR-IV channel memory. Let (p, q) be the *state transition* from state p to state q, and let $\pi_k(p)$ denote a *predecessor* for state p at time k, defined as the starting state associated with the *best* state transition. We

(A-1) Initialize $\Phi_0(p) = 0$ for $\forall p$

*(A-2) Initialize $\hat{\tau}_0(p) = 0$ and $\hat{\theta}_0(p) = 0$ for $\forall p$

(A-3) For $k = 0, 1, \ldots, L + \nu - 1$

(A-4) For $q = 0, 1, \ldots, Q - 1$

*(A-5) $y_k(p) = y(kT + \hat{\tau}_k(p))$ for $\forall p$

(A-6) $\rho_k(p, q) = |y_k(p) - \hat{r}(p, q)|^2$ for $\forall p$

(A-7) $\pi_{k+1}(q) = \arg\min_p\{\Phi_k(p) + \rho_k(p, q)\}$

(A-8) $\Phi_{k+1}(q) = \Phi_k(\pi_{k+1}(q)) + \rho_k(\pi_{k+1}(q), q)$

(A-9) $\mathbf{S}_{k+1}(q) = [\mathbf{S}_k(\pi_{k+1}(q)) \mid \pi_{k+1}(q)]$

*(A-10) $\hat{\epsilon} = \frac{3T}{16}\{y_k(\pi_{k+1}(q))\hat{r}(\pi_k(\pi_{k+1}(q)), \pi_{k+1}(q)) - y_{k-1}(\pi_k(\pi_{k+1}(q)))\hat{r}(\pi_{k+1}(q), q)\}$

*(A-11) $\hat{\theta}_{k+1}(q) = \hat{\theta}_k(\pi_{k+1}(q)) + \kappa\hat{\epsilon}$

*(A-12) $\hat{\tau}_{k+1}(q) = \hat{\tau}_k(\pi_{k+1}(q)) + \xi\hat{\epsilon} + \hat{\theta}_{k+1}(q)$

(A-13) End

(A-14) End

(A-15) Extract $\hat{\mathbf{a}}$ from the survivor path that minimizes $\Phi_{L+\nu}$

Figure 13: PSP-MM algorithm, where the lines beginning with * are the additional steps beyond the conventional Viterbi algorithm.

define $\hat{\tau}_k(p)$ as the k-th sampling phase offset for state p at time k, which is used to sample $y(t)$ at time k for the state transitions emanating from state p at time k, e.g., $y_k(p) = y(kT + \hat{\tau}_k(p))$, where $y_k(p)$ is the k-th sampler output for state p at time k. We also define $\hat{\theta}_k(p)$ as the k-th frequency error for state p at time k, which will be used to update $\hat{\tau}_k(p)$ (see (A-12)).

Consider the k-stage of the trellis. There are two state transitions arriving at state 2 at time $k + 1$, i.e., $(1, 2)$ and $(3, 2)$. We first sample $y(t)$ using $\hat{\tau}_k(1)$ and $\hat{\tau}_k(3)$ to obtain $y_k(1)$ and $y_k(3)$, respectively. Next, we compute two branch metrics $\rho_k(1, 2)$ and $\rho_k(3, 2)$ according to (A-6), where $\hat{r}(p, q)$ is the noiseless channel output associated with (p, q). Then, the starting state associated with the best state transition leading to state 2 at time $k + 1$ is chosen according to (A-7).

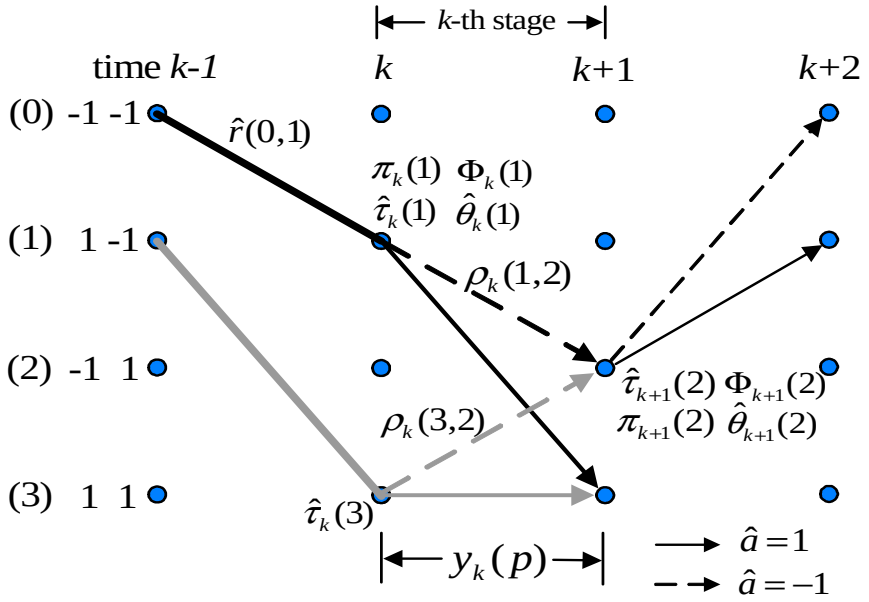


Figure 14: The PR-IV trellis structure explaining how PSP-MM performs.

Suppose $(1, 2)$ is the best state transition leading to state 2 at time $k+1$ so that $\pi_{k+1}(2) = 1$. The path metric for state 2 at time $k+1$, $\Phi_{k+1}(2)$, is updated based on (A-8), and the survivor path for state 2 at time $k+1$, $\mathbf{S}_{k+1}(2)$, is extended according to (A-9). Hence, we update the next sampling phase offset, $\hat{\tau}_{k+1}(2)$, based on (A-10) – (A-12) using the information from $\mathbf{S}_{k+1}(2)$. This $\hat{\tau}_{k+1}(2)$ will be employed to sample $y(t)$ at time $k+1$ for the state transitions emanating from state 2 at time $k+1$. We follow these steps according to the Viterbi algorithm for an entire received signal. Eventually, the decision is made by choosing the survivor path that has the minimum path metric.

Beyond the conventional Viterbi algorithm, PSP-MM needs new storage requirements for (i) the sampling phase offsets, (ii) the frequency errors, and (iii) the sampler outputs. Nevertheless, only those of the *current* and *previous* stages need to be stored, thus minimizing extra memory. Furthermore, it is evident that PSP-MM requires one PLL for each survivor path. Consequently, for a PR-IV channel with four states, the complexity of timing recovery of PSP-MM is four times the complexity of conventional

timing recovery.

3.3.2 New Timing Error Detector

The PSP-MM described in Section 3.3.1 does not exploit the *future* information available in the trellis, i.e., the channel output at the next time instant. Consider the case where we are at state 2 at time $k+1$, we would know exactly that there will be two state transitions emanating from this state, i.e., $(2,0)$ and $(2,1)$. Since the two future channel outputs, $\hat{r}(2,0)$ and $\hat{r}(2,1)$, are available at time k, it might be a good idea to incorporate them for the timing update operation at time k.

In doing so, we need to develop the TED algorithm that is able to use future information. One such TED algorithm can be found by minimizing the log-likelihood function of the samples $\{y_k\}$ according to [48]

$$\begin{aligned} L(\mathbf{y}|\hat{\mathbf{r}},\epsilon) &= \sum_m \left| y_m - \sum_n \hat{r}_n q((m-n)T - \tau + \hat{\tau}) \right|^2 \\ &= A - \sum_m y_m \sum_n \hat{r}_n q((m-n)T - \tau + \hat{\tau}), \end{aligned} \tag{19}$$

where A is a constant independent of $\hat{\tau}$, $y_m = y(mT+\hat{\tau})$, τ is the actual timing offset, and $\hat{\tau}$ is an estimate of τ.

Since we are concerned with an error feedback algorithm, only $\hat{\tau}$ close to τ is of interest [46, 48]. Thus, the timing error signal can be obtained by differentiating (19) with respect to $\hat{\tau}$, i.e.,

$$\frac{\partial L(\mathbf{y}|\hat{\mathbf{r}},\epsilon)}{\partial \hat{\tau}} = -\sum_m y_m \sum_n \hat{r}_n \dot{q}((m-n)T), \tag{20}$$

where $\dot{q}(kT)$ is the derivative of $q(t)$ evaluated at time kT, which can be expressed as

$$\dot{q}(kT) = \begin{cases} 0 & k = 0 \\ \frac{1}{kT}(-1)^k & \text{otherwise} \end{cases}. \tag{21}$$

With a symmetric property [48], the estimated timing error at time k for four observations can be written as

$$\hat{\epsilon}_k = -\frac{180T}{3086} \sum_{m=k-2}^{k+1} y_m \sum_{n=k-2}^{k+1} \hat{r}_n \dot{q}((m-n)T)$$

$$= \frac{180T}{3086}\{y_{k+1}(\hat{r}_k - 0.5\hat{r}_{k-1} + \hat{r}_{k-2}/3) + y_k(-\hat{r}_{k+1} + \hat{r}_{k-1} - 0.5\hat{r}_{k-2}) + y_{k-1}(0.5\hat{r}_{k+1} - \hat{r}_k + \hat{r}_{k-2}) + y_{k-2}(-\hat{r}_{k+1}/3 + 0.5\hat{r}_k - \hat{r}_{k-1})\}, \quad (22)$$

where we approximate $y_{k+1} = y(kT+T+\hat{\tau}_k)$ assuming that the timing offset is slowly varying. The constant $180T/3086$ is introduced to ensure that the S-curve slope of (22) is unity at the origin. We will refer to this TED as "4S-TED," where "4S" stands for the number of samples taken from time $k-2$ to $k+1$ that are used to compute $\hat{\epsilon}_k$. It should be noted that when using the samples only at time $k-1$ and k, (22) reduces to the M&M TED in (4) (by ignoring the constant term of both TEDs).

It is worth exploring the characteristics of both TEDs, which can be determined by the timing function or S-curve, as described in Section 2.3.3. For a PR-IV channel, the timing function of the M&M TED is given by (16), whereas that of the 4S-TED can be expressed as

$$\begin{aligned} S_{\text{TED}}(\epsilon) &= E[\hat{\epsilon}_k \,|\, \epsilon, \hat{r}_{k-2} = r_{k-2}, \hat{r}_{k-1} = r_{k-1}, \hat{r}_k = r_k, \hat{r}_{k+1} = r_{k+1}] \\ &= \frac{180T}{3086}\{-h(-\epsilon) + 6h(T-\epsilon) - h(2T-\epsilon) \\ &\quad - h(-3T-\epsilon)/3 + h(-2T-\epsilon) - 8h(-T-\epsilon)/3 \\ &\quad - 8h(3T-\epsilon)/3 + h(4T-\epsilon) - h(5T-\epsilon)/3\}. \end{aligned} \quad (23)$$

The mean and the standard deviation of both TEDs (when employed in PSP-based timing recovery) as a function of normalized timing errors ϵ/T's at $E_b/N_0 =$ 10 dB are plotted in Figure 15, assuming that we have access to the correct future information. Clearly, both timing functions are odd symmetric with respect to $\epsilon = 0$. Thus, regardless of the TED used, the sampling phase offset updated according to (18) will settle down in the steady-state at $\epsilon = 0$. Observe that the mean of both TEDs is approximately proportional to ϵ/T over a range of $\pm 20\%$ about the origin. As expected, the standard deviation of the 4S-TED is lower than that of the M&M TED because more information is used in evaluating the estimated timing error. Therefore,

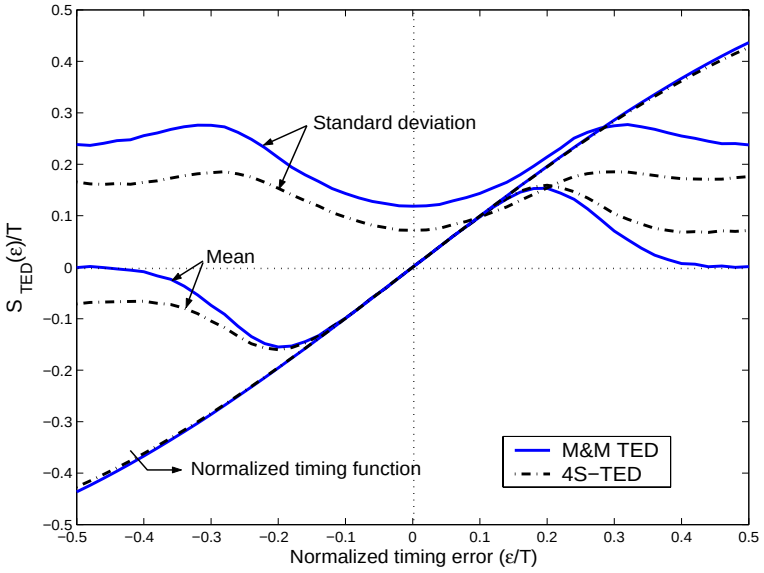


Figure 15: The mean and the standard deviation of different TEDs used in PSP-based timing recovery for a PR-IV channel at $E_b/N_0 = 10$ dB.

the 4S-TED is more robust to the noise in the timing error signal than the M&M TED.

Unlike PSP-MM, for a PR-IV channel with four states, the complexity of timing recovery of PSP-based timing recovery with 4S-TED is eight times that of conventional timing recovery because it requires one PLL for each state transition in one stage of the trellis.

3.3.3 Note on Conventional Timing Recovery

We can also explain how conventional timing recovery works in the context of the trellis structure. This will show that it is in fact a special case of PSP-based timing recovery.

Practically, conventional timing recovery employs the *same* sampling phase offset $\hat{\tau}_k$ to sample $y(t)$ for all state transitions at time k. Then, the *same* decision (either the hard decision $\hat{r}_k$ or the tentative decision $\hat{r}_{k-d}$ found by tracing back d steps along the best survivor path chosen at time k) is used to compute the estimated timing error

$\hat{\epsilon}_k$ for all states, which finally results in the *same* $\hat{\tau}_{k+1}$ for all states after updating it.

Therefore, PSP-based timing recovery differs from conventional timing recovery in the sense that (i) it uses different sampling phase offsets associated with each state transition to sample $y(t)$; and (ii) it employs the instantaneous decision with zero decision delay associated with each state transition to compute the estimated timing error.

3.4 Numerical Results and Discussion

In simulation, unless otherwise specified, we consider $\sigma_w/T = 0.5\%$ and employ the PLL gain parameter, ξ, designed to recover phase change within $C = 100$ bit periods, based on a linearized model of first-order PLL, and assuming that the S-curve slope is unity at the origin and there is no noise in the system. The ξ's designed for the decision delays of $0, 4T, 8T$, and $20T$ are 0.030, 0.027, 0.025, and 0.019, respectively.

We first explore how the decision delay affects the performance of timing recovery. In doing so, we consider the PSP-MM scheme, where we have access to all the decisions $\{\hat{r}_{k-d}\}$ (at any d steps earlier) associated with each survivor path. Figure 16(a) compares the performance of different PSP-MMs, where the RMS timing error σ_ϵ/T is plotted as a function of E_b/N_0's. Apparently, PSP-MM with $d = 0$ yields the best performance. This can be confirmed by plotting the σ_ϵ/T performance as a function of ξ's at $E_b/N_0 = 8$ dB in Figure 16(b). Again, PSP-MM with $d = 0$ performs better than PSP-MM with $d \neq 0$ for all ξ's. Results imply that the decision delay has a tremendous impact on overall performance. Consequently, it is desirable to use the decision with zero decision delay whenever possible.

Figure 16 also shows the performance of PSP-based timing recovery with 4S-TED for $d = 0$. As shown in Figure 16(a), PSP-based timing recovery with 4S-TED performs better than PSP-MM at low E_b/N_0's. This might be because the future information used in the 4S-TED helps improve the performance of timing

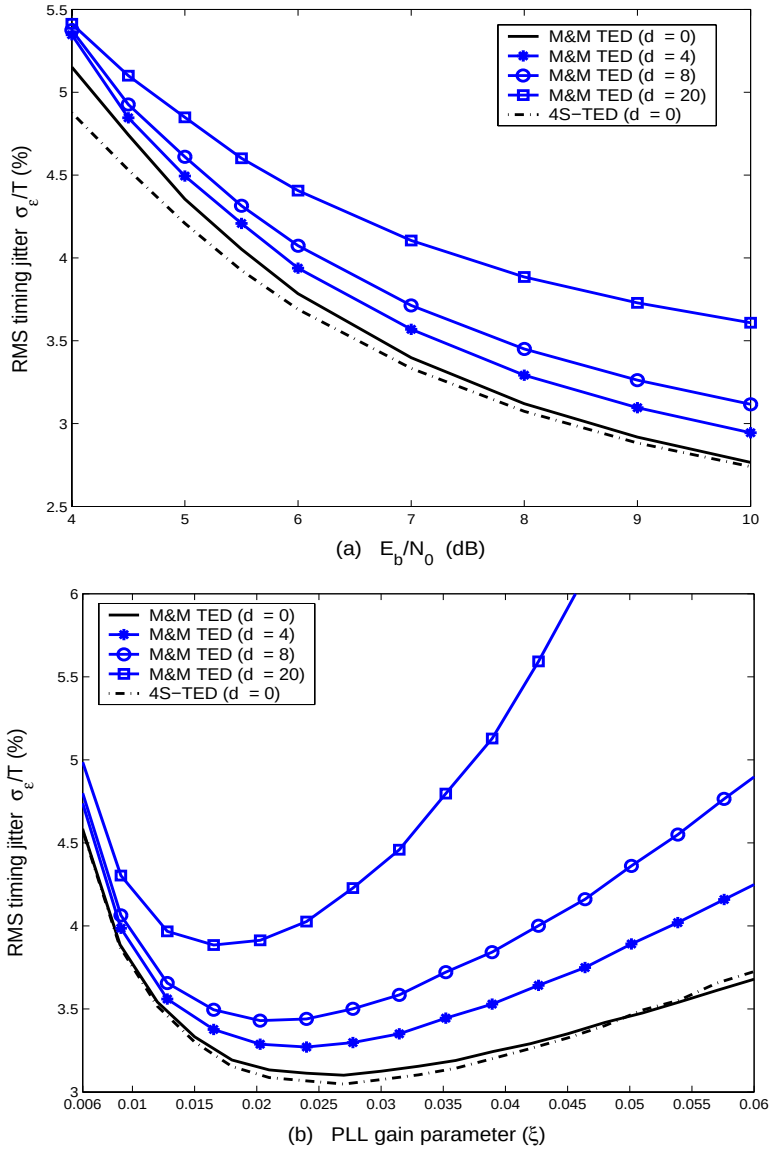


Figure 16: Performance comparison of PSP-based timing recovery with different TEDs (a) as a function of E_b/N_0's and (b) as a function of ξ's at $E_b/N_0 = 8$ dB.

recovery when uncertainty is high. Figure 16(b) also indicates that PSP-based timing recovery with 4S-TED yields slightly lower σ_ϵ/T performance than PSP-MM for small ξ. However, it starts performing worse than PSP-MM when ξ is large. Since PSP-based timing recovery with 4S-TED provides only a small gain over PSP-MM but its complexity is much higher than the complexity of PSP-MM, PSP-MM is then preferred. From this point on, we will consider only PSP-MM with $d = 0$ when

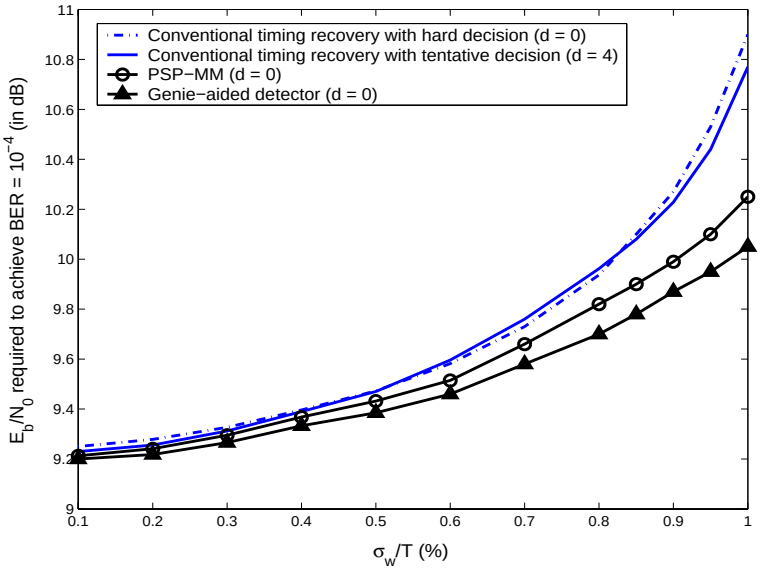


Figure 17: Performance comparison of different timing recovery schemes.

comparing the performance of PSP-based timing recovery with conventional timing recovery.

Finally, we compare the performance of PSP-MM with that of conventional timing recovery by plotting the E_b/N_0 (in dB) requirement for BER $= 10^{-4}$ as a function of σ_w/T's in Figure 17. The curve labeled "a genie-aided detector" represents conventional timing recovery whose PLL has access to all correct decisions, thus serving as a lower bound for a timing recovery scheme that is based on a PLL. Obviously, PSP-MM performs better than conventional timing recovery, especially when σ_w/T is large. As shown in Figure 17, PSP-MM is 0.5 dB better than conventional timing recovery when operating in a system with $\sigma_w/T = 1\%$. Although conventional timing recovery with hard decision seems to perform comparably to that with tentative decision, this is not true when the channel is complex, e.g., the channel with large memory or with arbitrary coefficients. That is why conventional timing recovery practically utilizes the tentative decision provided by the Viterbi detector in most applications.

The reason that PSP-MM performs better than conventional timing recovery can be intuitively explained as follows. At each time instant, at least one state transition

in each trellis stage will correspond to the correct decision. Using that decision to perform the timing update operation will then improve the performance of timing recovery. In other words, the PLL is *fully trained* if the correct path is chosen. By following this idea for an entire received signal, the overall system performance will be improved.

3.5 Convergence Behavior

To investigate the convergence behavior of different timing recovery schemes, we run another experiment but this time we look at the timing estimate. Consider the same perfectly equalized PR-IV channel model shown in Figure 4 with $\sigma_w/T = 0\%$; however, a huge timing offset $\hat{\tau}_0 = 0.5T$ is used. We design a PLL gain parameter ξ based on a linearized model of PLL for $C = 50$, where the ξ's designed for the decision delays of 0 and $4T$ are 0.058 and 0.049, respectively.

Figure 18 shows the timing estimate behavior of different timing recovery schemes at $E_b/N_0 = 10$ dB based on 50 data packets. As expected, the timing estimate curve of a genie-aided detector converges within 50 bit periods. This is because the timing update operation is always performed on the correct decisions. On the other hand, other schemes do not converge to the desired value $\tau = 0$ or T within 50 bit periods because the decisions used in the timing update operation depend on the noise level embedded in the received analog signal. Nonetheless, as depicted in Figure 18, it seems that PSP-MM converges faster than conventional timing recovery.

To justify that PSP-MM achieves faster convergence than conventional timing recovery, we plot the percentage of convergence versus time (in bit periods) of different timing recovery schemes based on 50000 data packets in Figure 19, where the convergence is achieved at time k when $\hat{\tau}_i$, for $i \geq k$, is equal to the desired value 0 or T with $\pm 10\%$ tolerance. It is evident that the genie-aided detector converges within 50 bit periods. Furthermore, conventional timing recovery with tentative decision converges

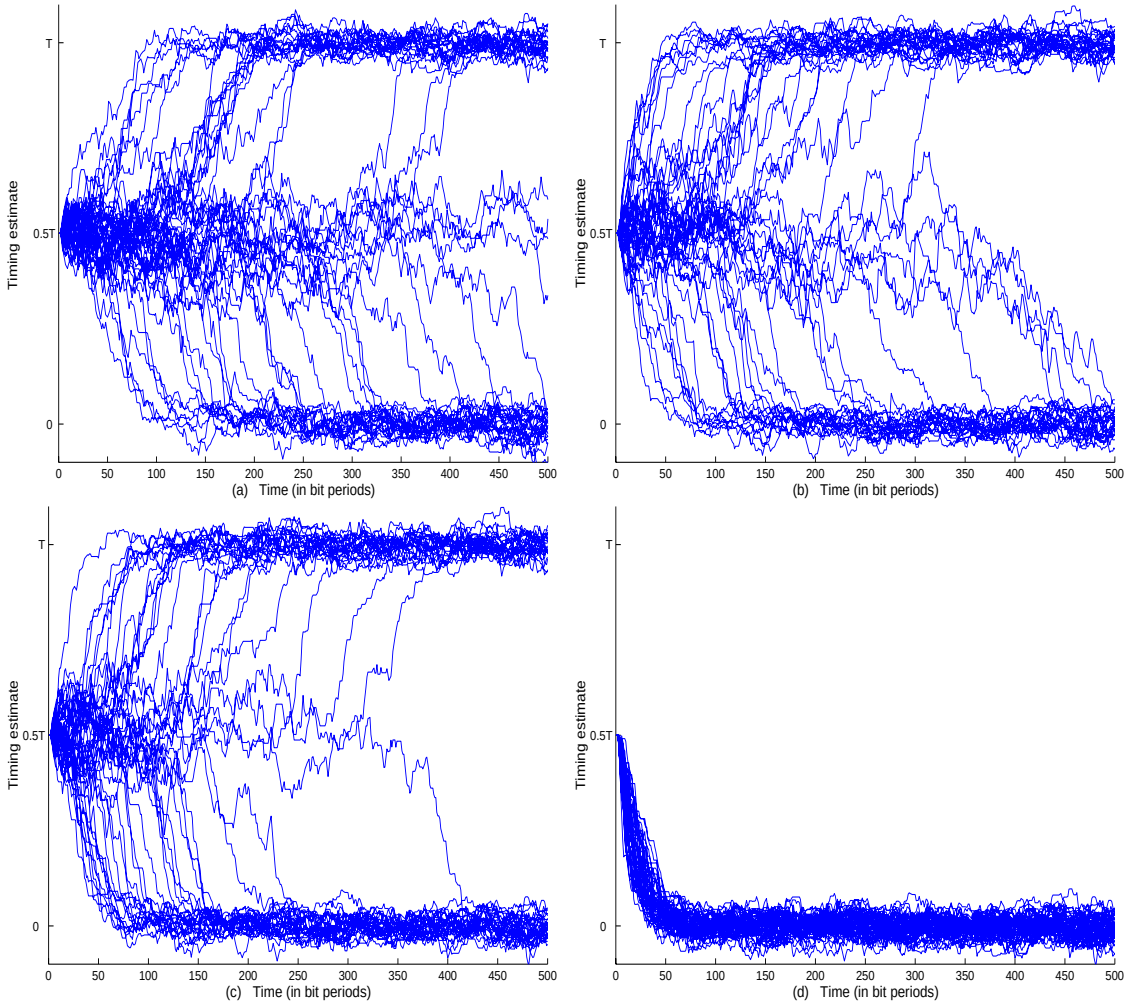


Figure 18: Timing estimate plots of (a) conventional timing recovery with hard decision ($d = 0$), (b) conventional timing recovery with tentative decision ($d = 4$), (c) PSP-MM ($d = 0$), and (d) a genie-aided detector ($d = 0$), at $E_b/N_0 = 10$ dB based on 50 runs.

faster than that with hard decision. This is because the tentative decision is more reliable than the hard decision. This explains why conventional timing recovery practically uses the Viterbi detector with a short decision delay as the symbol detector in most applications [15]. As expected, PSP-MM achieves faster convergence than conventional timing recovery. This can be implied that PSP-MM will perform better than conventional timing recovery when operating in a system that requires fast convergence. We can verify this statement by plotting the BER performance of different

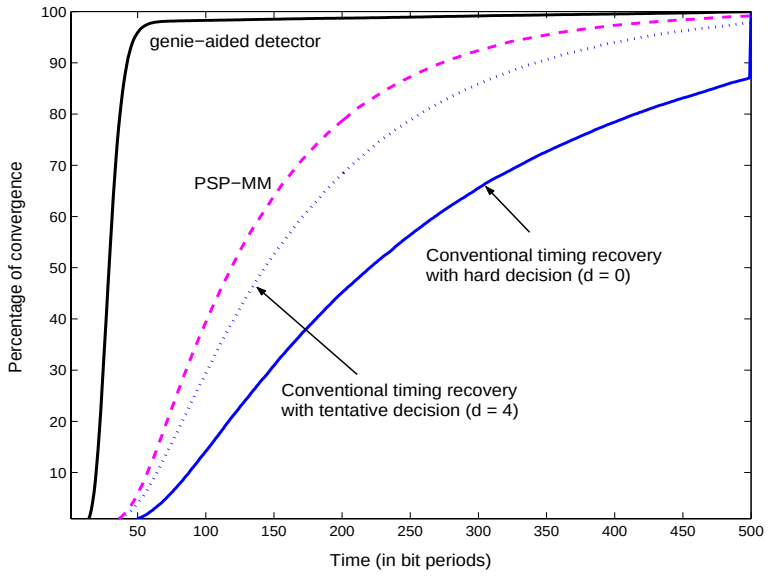


Figure 19: Percentage of convergence of different timing recovery schemes at E_b/N_0 = 10 dB based on 50000 runs.

timing recovery schemes in Figure 20, using the PLL gain parameters designed for different C's when operating in a moderate system condition, e.g., with $\sigma_w/T = 0.5\%$ and 0.2% frequency offset. Clearly, a large performance gap between PSP-MM and conventional timing recovery can be obtained at high E_b/N_0, especially when C is small.

3.6 Reduced-Complexity PSP-Based Timing Recovery

Because PSP-MM is developed based on the Viterbi algorithm, its complexity grows exponentially with channel memory [24]. There are several approaches proposed in the literature to reduce the complexity of the Viterbi algorithm, including reduced-state sequence estimation (RSSE) [22], delayed decision-feedback sequence estimation (DDFSE) [19], the M-algorithm [72], and the T-algorithm [73]. However, it has been shown in [68, 70, 74] that the M- and T-algorithms seem to be more efficient than RSSE and DDFSE, when the suitable values for their parameters are chosen.

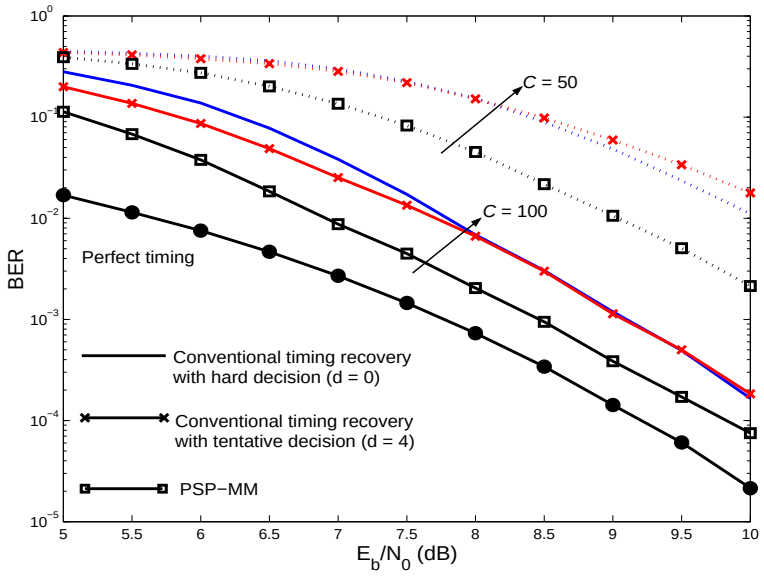


Figure 20: BER performance of different timing recovery schemes using the PLL gain parameters designed for different C's for systems with $\sigma_w/T = 0.5\%$ and 0.2% frequency offset.

Consequently, in this section, we focus on reducing the complexity of PSP-MM based only on the M- and T-algorithms, and then investigate their performances.

3.6.1 The M-algorithm

The M-algorithm, a breadth-first trellis search algorithm, was first introduced by Simmons and Mohan [72]. It has been employed in many applications, including source coding [72] and channel decoding [70]. It performs in a same manner as the Viterbi algorithm does, except with an additional discard rule, which can be explained as follows.

At each time instant, the M-algorithm first finds the *minimum* path metric leading to each trellis state. Hence, it retains only the M (M must be less than the total number of states in one stage of the trellis) paths with the *lowest* path metrics among all survivor paths. Therefore, the M-algorithm performs the same number of computational operations at each time instant.

3.6.2 The T-algorithm

The T-algorithm has been proposed by Simmons [73]. It is also a breadth-first trellis search algorithm but it has a different discard rule. At each time instant, it first finds the best overall *minimum* path metric, Φ_B. Then, it discards all paths whose path metrics are *larger* than Φ_B by a threshold value, T, in percentage of Φ_B. Consequently, the T-algorithm has a varying number of paths kept at each time instant.

3.6.3 Performance Comparison

To compare the performance of PSP-MM with different reduced-complexity approaches, we consider the BER and the average number of searched states (or survivor paths) [74]. The latter provides the amount of required computational time and determines the memory requirement because path extension and update operations per path are identical for all approaches. It should be noted that the M-algorithm has an additional sorting overhead at each time instant, and the T-algorithm requires a storage capacity equal to the total number of states in one trellis stage because it has a varying number of paths kept at each time instant, which will slow down the execution time [68]. This excess requirement must be taken into consideration when comparing the overall system performance.

Figure 21 compares the performance of PSP-MM with different reduced-complexity approaches. Apparently, there is a trade-off between the BER and the average number of searched states. That is, the larger the average number of searched states, the lower the BER. At low E_b/N_0, the M-algorithm does not perform well, which can be possibly explained as follows. When erroneous equalization has occurred, it causes many values of the path metrics to have roughly similar values, instead of the usual wide spread of values. Thus, the M-algorithm is more likely to discard the correct path. On the contrary, the T-algorithm is guided by the size of the path metrics. Hence, when there are many values of significant size, the T-algorithm expands the

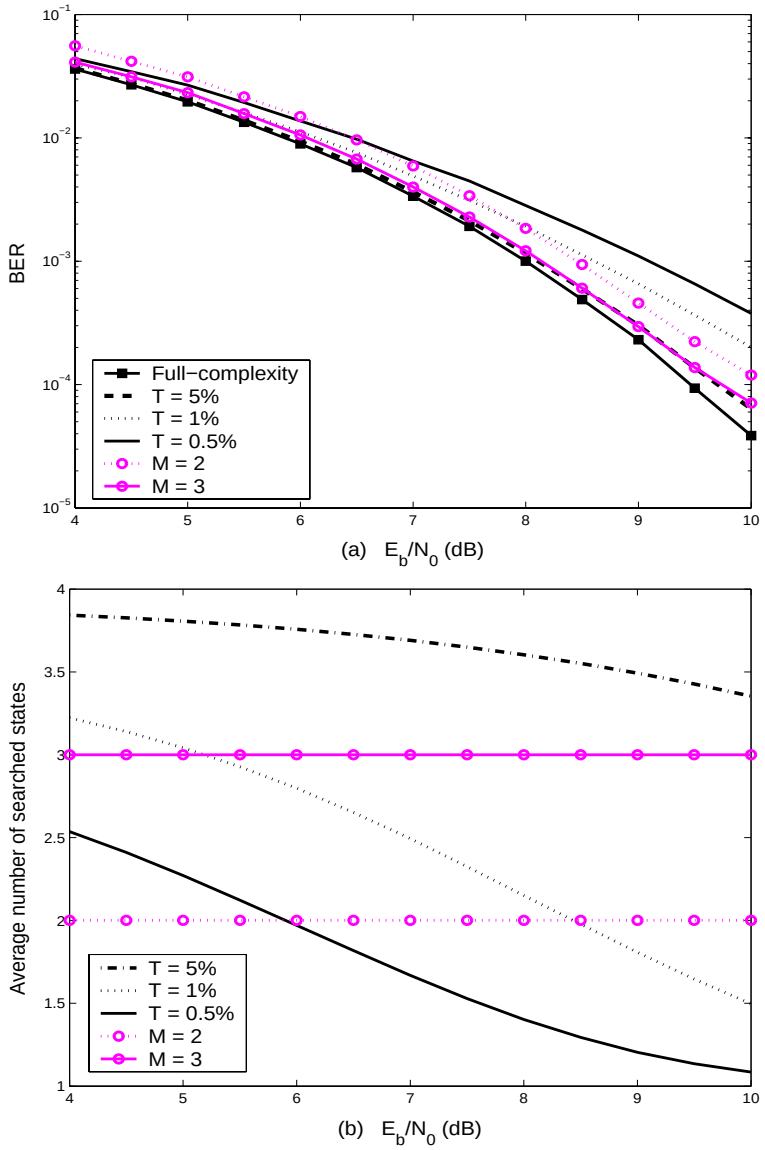


Figure 21: (a) BER and (b) the average number of search states performances as a function of E_b/N_0's of PSP-MM with different reduced-complexity approaches for a PR-IV channel with $\sigma_w/T = 0.5\%$.

computation to include them all. For this specific channel model, although the M-algorithm with M = 3 and the T-algorithm with T = 5% are comparable in terms of BER, the M-algorithm is preferred because it requires a fewer number of searched states than the T-algorithm. Note that we also observed that a similar result is obtained when operating in a system with any σ_w/T.

3.7 Summary

We proposed PSP-based timing recovery for PR channels to improve the performance of conventional timing recovery without exploiting the presence of error-correction codes. The proposed scheme jointly performs timing recovery and equalization.

It is apparent that the delay in the timing loop has a dominant impact on overall performance. That is why PSP-MM with $d = 0$ performs better than PSP-MM with $d \neq 0$. Therefore, PSP-based timing recovery has the advantage of reducing the delay in the timing loop. Since PSP-based timing recovery with 4S-TED provides only a small gain over PSP-MM, PSP-MM is then preferred because it has less complexity than PSP-based timing recovery with 4S-TED.

We have shown that PSP-MM yields better performance than conventional timing recovery, especially when the timing jitter is large. Specifically, PSP-MM provides a 0.5 dB gain over conventional timing recovery when operating in a system with $\sigma_w/T = 1\%$. It should be pointed out that as the complexity of PSP-based timing recovery is high, all advantages obtained from PSP-based timing recovery must be balanced against the increased implementation cost.

We also investigated the convergence behavior of different timing recovery schemes. It has been shown that PSP-based timing recovery can achieve faster convergence than conventional timing recovery. This explains why PSP-based timing recovery performs better than conventional timing recovery when operating in a system that requires fast convergence (i.e., when using the PLL gain parameters designed for small C).

Finally, we have investigated the performance of a reduced-complexity version of PSP-based timing recovery. We found that the M- and T- algorithms can be used to reduce the complexity of PSP-based timing recovery with acceptable performance if their parameters are suitably chosen. Apparently, there is a fundamental trade-off between the complexity and the BER performance.

CHAPTER 4

PER-SURVIVOR ITERATIVE TIMING RECOVERY FOR CODED PARTIAL RESPONSE CHANNELS

This chapter deals with the problem of timing recovery operating at low signal-to-noise ratio (SNR). In this case, we consider a coded system because a large coding gain of iterative error-correction codes (ECCs) allows reliable operation at low SNR. A new iterative timing recovery scheme based on PSP is proposed, and its performance is compared with conventional schemes. Then, a reduced-complexity version of the proposed scheme is given and investigated. Finally, we propose to use the exit transfer information chart (EXIT chart) analysis as a tool to compare and predict the performance of iterative timing recovery schemes because the BER computation takes a considerable amount of simulation time.

4.1 Introduction

Iterative ECCs, such as turbo codes [10] and low-density parity-check (LDPC) codes [26], allow reliable operation at low SNR because of their large coding gains [10, 30, 86]. Furthermore, the principle of iterative decoding can also be extended to include equalization, which is commonly known as *turbo equalization* [67, 79]. This means that timing recovery must also function at low SNR. In practice, a conventional receiver performs timing recovery and error-correction decoding separately. Specifically, conventional timing recovery ignores the presence of ECCs. Thus, it fails to work properly at low SNR (as will be seen later in simulation results).

Theoretically, joint ML estimation of timing offsets and message bits, which will jointly perform timing recovery, equalization, and decoding, is a preferred method of synchronization [48] but its complexity is huge. A solution based on the expectation-maximization (EM) algorithm [27, 57] is also complex. Fortunately, a solution to this problem with complexity comparable to the conventional receiver has been proposed by Nayak, Barry, and McLaughlin [56], which will be referred to as the NBM scheme. It is realized by embedding the timing recovery step inside the turbo equalizer [67, 75] so as to perform timing recovery, equalization, and error-correction decoding jointly. Nonetheless, this scheme requires a large number of turbo iterations to provide a good performance even with a cycle slip [9] detection and correction algorithm as used in [56], especially when the timing jitter is severe.

As discussed in Section 3, we applied PSP to develop PSP-based timing recovery, which is implemented based on a Viterbi algorithm. Similarly, we can also perform timing recovery and equalization jointly based on a BCJR algorithm [5]. To do so, we apply the PSP concept to the BCJR algorithm, resulting in a per-survivor BCJR equalizer, denoted as "PSP-BCJR." Hence, we propose a *per-survivor iterative timing recovery* scheme [35], which iteratively exchanges soft information between PSP-BCJR and a soft-in soft-out (SISO) decoder. Although each iteration of per-survivor iterative timing recovery has high complexity, it can automatically correct a cycle slip much more efficiently than the NBM scheme. In other words, per-survivor iterative timing recovery requires fewer turbo iterations than the NBM scheme to yield a good performance.

4.2 Prior Work

Although we *independently* develop the PSP-BCJR module in this work, it turns out that PSP-BCJR shares many similarities with the so-called *adaptive SISO* module

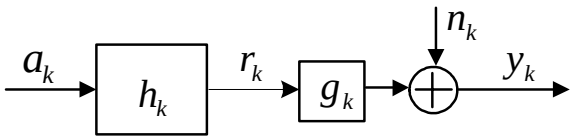


Figure 22: An equivalent discrete-time channel model.

developed by Anastasopoulos and Chugg [1, 2, 14]. Before demonstrating that PSP-BCJR can be considered as a special case of the adaptive SISO module, it is worth briefly describing the adaptive SISO algorithm.

The adaptive SISO algorithm was proposed to deal with unknown parameters in iterative detection. The *exact* expression of soft information, produced by the adaptive SISO algorithm, in the presence of parametric uncertainty can be explained as follows.

4.2.1 Channel Model

Consider the equivalent discrete-time channel model shown in Figure 22, where $a_k \in \{\pm 1\}$ is a binary input sequence, h_k is the channel impulse response, r_k is the noiseless channel output, g_k is the unknown parameter, and n_k is an *i.i.d.* zero-mean Gaussian random variable with variance σ_n^2. The observation, y_k, can then be written as

$$y_k = \mathbf{r}_k^{\mathrm{T}} \mathbf{g}_k + n_k, \tag{24}$$

where $\mathbf{r}_k$ and $\mathbf{g}_k$ are the column vectors of $\{r_k\}$ and $\{g_k\}$ at time k, respectively, and $(\cdot)^{\mathrm{T}}$ is the transpose operation. To derive the exact expression of soft information, it is *assumed* [2] that the column vector process $\mathbf{g}_k$ is a first-order Gaussian Markov (GM) process [2] and stationary according to

$$\mathbf{g}_k = \mathbf{G}\mathbf{g}_{k-1} + \mathbf{w}_k, \tag{25}$$

where $\mathbf{w}_k$ is the zero-mean Gaussian vector with covariance $\mathbf{K}_w(i,j) = \mathbf{Q}\delta(i,j)$, $\delta(i,j)$ is the Kronecker delta function [42], and the covariance matrix of $\mathbf{g}_k$, $\mathbf{K}_g$, must satisfy

[2]

$$\mathbf{K}_g = \mathbf{G}\mathbf{K}_g\mathbf{G}^{\mathrm{T}} + \mathbf{Q}. \tag{26}$$

Under this condition, its time-reversed process, $\mathbf{g}_{-k}$, is also a first-order GM process, i.e.,

$$\mathbf{g}_k = \mathbf{G}^b\mathbf{g}_{k+1} + \mathbf{v}_k, \tag{27}$$

where [2]

$$\begin{aligned} \mathbf{G}^b &= \mathbf{K}_g\mathbf{G}^{\mathrm{T}}\mathbf{K}_g^{-1}, \\ \mathbf{K}_v(i,j) &= \mathbf{Q}^b\delta(i,j), \\ \mathbf{Q}^b &= \mathbf{K}_g - \mathbf{K}_g\mathbf{G}^{\mathrm{T}}\mathbf{K}_g^{-1}\mathbf{G}\mathbf{K}_g. \end{aligned}$$

4.2.2 Optimal Adaptive SISO Algorithm

The objective of the SISO algorithm is to generate the soft information about the input and the output symbols of the channel based on the observations. Denote $y_i^m = [y_i\, y_{i+1}\, \cdots\, y_{m-1}\, y_m]$ as a collection of the samples from time i to time m. For a generic quantity u_k (i.e., a_k or r_k), the soft information of the form

$$\mathrm{APP}(u_k) = \Pr[u_k|y_0^L] = c\sum_{a_0^L:u_k}\Pr[y_0^L, a_0^L] = c\sum_{a_0^L:u_k} E_{g_k}[\Pr[y_0^L, a_0^L|g_k]] \tag{28}$$

is considered in [2], where $a_0^L : u_k$ denotes all possible input sequences consistent with u_k, $L+1$ is the length of the input sequence, and c is a normalized constant.

A close form solution of (28) can be easily obtained by decomposing $\Pr[y_0^L, a_0^L]$ into three terms according to [2]

$$\begin{aligned} \Pr[y_0^L, a_0^L] &= \underbrace{\Pr[y_0^k, a_0^k]}_{\text{past/present}} \cdot \underbrace{\Pr[y_{k+1}^L, a_{k+1}^L|\Psi_{k+1}]}_{\text{future}} \\ &\quad \cdot \underbrace{\int_{g_k} \frac{\Pr[g_k|y_0^k, a_0^k]\cdot\Pr[g_k|\Psi_{k+1}, y_{k+1}^L, a_{k+1}^L]}{\Pr[g_k]}\, dg_k}_{\text{binding factor } B_f}, \end{aligned} \tag{29}$$

where Ψ_k is the state in the *tree* at time k. The first term in (29) depends on past and present information, which can be computed recursively as [31]

$$\begin{aligned}\Pr[y_0^k, a_0^k] &= \Pr[y_k|y_0^{k-1}, a_0^{k-1}] \cdot \Pr[a_k] \cdot \Pr[y_0^{k-1}, a_0^{k-1}] \\ &= \mathcal{N}(y_k;\, \mathbf{r}_k^{\mathrm{T}} \tilde{\mathbf{g}}_{k|k-1};\, \sigma_n^2 + \mathbf{r}_k^{\mathrm{T}} \tilde{\mathbf{G}}_{k|k-1} \mathbf{r}_k) \cdot \Pr[a_k] \cdot \Pr[y_0^{k-1}, a_0^{k-1}], \quad (30)\end{aligned}$$

where $\mathcal{N}(y; \mu; \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(y-\mu)^2}{2\sigma^2})$ is a Gaussian distribution function with mean μ and variance σ^2, and $\tilde{\mathbf{g}}_{k|k-1}$ and $\tilde{\mathbf{G}}_{k|k-1}$ are the one-step *forward* channel predictor and its corresponding covariance generated by a Kalman filter (KF) [31]. Similarly, the second term, relying only on the future information, can also be calculated recursively by [2]

$$\begin{aligned}\Pr[y_{k+1}^L, a_{k+1}^L|\Psi_{k+1}] &= \Pr[y_{k+1}|y_{k+2}^L, a_{k+1}^L, \Psi_{k+1}] \cdot \Pr[a_{k+1}] \cdot \Pr[y_{k+2}^L, a_{k+2}^L|\Psi_{k+2}] \\ &= \mathcal{N}(y_{k+1};\, \mathbf{r}_{k+1}^{\mathrm{T}} \tilde{\mathbf{g}}_{k+1|k+2}^b;\, \sigma_n^2 + \mathbf{r}_{k+1}^{\mathrm{T}} \tilde{\mathbf{G}}_{k+1|k+2}^b \mathbf{r}_{k+1}) \\ &\quad \cdot \Pr[a_{k+1}] \cdot \Pr[y_{k+2}^L, a_{k+2}^L|\Psi_{k+2}], \quad (31)\end{aligned}$$

where $\tilde{\mathbf{g}}_{k|k+1}^b$ and $\tilde{\mathbf{G}}_{k|k+1}^b$ are the one-step *backward* channel predictor and its corresponding covariance generated by a KF. The binding factor, B_f, (i.e., the third term in (29)) measures the dependency of the past, present, and future information that is introduced by $\mathbf{g}_k$ and would be eliminated in the absence of parametric uncertainty. Note that, for $\mathbf{g}_k$ being *only* a GM process and stationary, the closed form of B_f can be derived, as given in [2] (see Appendix in [2]).

Figure 23 illustrates how (28) is evaluated, which can be explained as follows. Starting at time 0, a forward 2-ary tree is built and each node represents a valid sequence path. Hence, $\Pr[y_0^k, a_0^k]$, $\tilde{\mathbf{g}}_{k|k-1}$, and $\tilde{\mathbf{G}}_{k|k-1}$, which are associated with each forward path, are computed and stored at each node. Similarly, starting at time L, a backward 2-ary tree is built and $\Pr[y_{k+1}^L, a_{k+1}^L|\Psi_{k+1}]$, $\tilde{\mathbf{g}}_{k|k+1}^b$, and $\tilde{\mathbf{G}}_{k|k+1}^b$, which are associated with each backward path, are computed and stored at each node. After k forward and $L-k$ backward steps, the two trees meet each other. The 2^{k+1} likelihoods

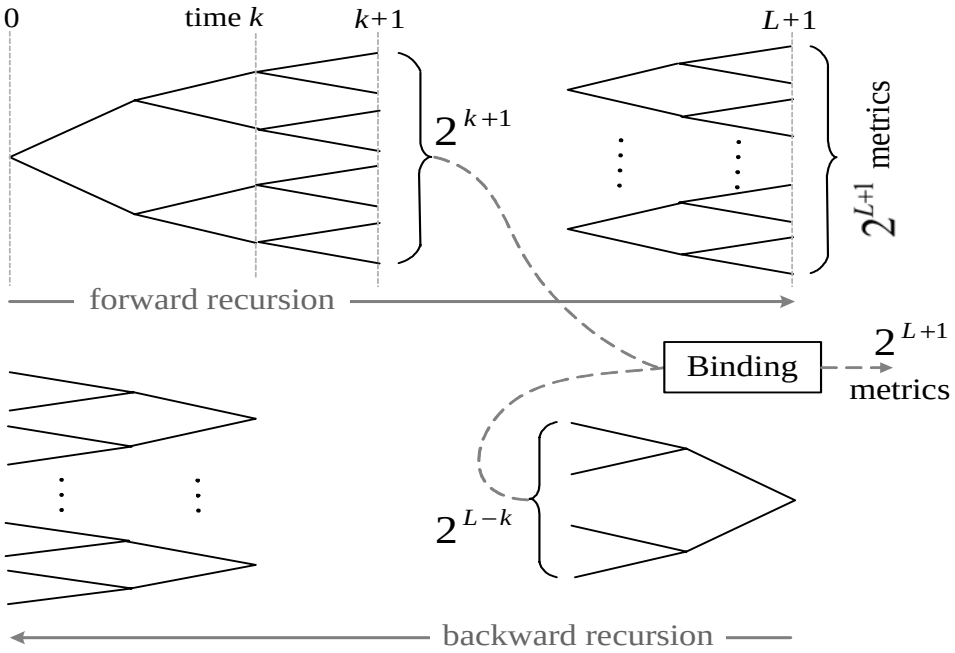


Figure 23: Likelihood computation for forward and backward recursions [1].

corresponding to the nodes of the forward tree are combined with the 2^{L-k} likelihoods corresponding to the nodes of the backward tree, and then weighted by the binding factor B_f. Finally, the soft output for u_k is produced by computing the summation over all terms that correspond to u_k.

4.2.3 Suboptimal Adaptive SISO Algorithm

As shown in (29), the exact expression of soft information involves likelihood updates on both the forward and the backward trees with the aid of per-path KFs, followed by binding of the past and future metrics. This implies that the exact expression has very high complexity. To reduce its complexity, some possible simplifications have been suggested in [2], including (i) using a non-exhaustive tree search, (ii) using a non-Kalman parameter estimator, and (iii) using a suboptimal binding factor. For example, the exhaustive tree search can be avoided by employing the PSP technique [65] so that the KF parameter estimate can be kept only for every trellis state and

updated in a PSP fashion. The second simplification is to replace a KF parameter estimator with a simple estimator, e.g., the estimator that is based on the least mean-squared (LMS) algorithm [8]. The last simplification is to replace the optimal binding factor (see Appendix in [2]) with a suboptimal one [1, 2, 14].

Based on all possible simplifications, several suboptimal adaptive SISO algorithms can be realized. Some of them have been investigated in the applications of the trellis-coded modulation (TCM) in interleaved frequency-selective fading channels [2] and in the turbo-coded systems with carrier phase tracking [3]. Interestingly, we found that one of the suboptimal adaptive SISO algorithms is the *same* as the PSP-BCJR module if the following happens (see Appendix A in [3]): (i) a PSP technique is used instead of an exhaustive tree search; (ii) the phase error detector is replaced by an M&M TED; and (iii) the binding factor is ignored.

4.2.4 Note on the Adaptive SISO Algorithm

For the application of timing recovery in the ISI channel with time-varying timing offsets, the exact expression of soft information given in (29) is no longer valid. To justify this statement, we consider the system model shown in Figure 4. By assuming that $y(t)$ is bandlimited, we can represent $y(t)$ with its corresponding set of samples $\{y_k\}$, where

$$y_k = \sum_{i=k-m}^{k+m} r_i q(kT - iT - \tau_i) + n_k. \tag{32}$$

This is because the bandlimited nature of $y(t)$ makes $\{y_k\}$ sufficient statistics. Equation (32) can also be expressed in a matrix form, similar to (24), as

$$y_k = \mathbf{r}_k^{\mathrm{T}} \mathbf{g}_k + n_k, \tag{33}$$

but with

$$\mathbf{g}_k = \begin{bmatrix} q(mT - \tau_{k-m}) \\ \vdots \\ q(-\tau_k) \\ \vdots \\ q(-mT - \tau_{k+m}) \end{bmatrix}, \tag{34}$$

and $\mathbf{r}_k = [r_{k-m} \cdots r_k \cdots r_{k+m}]^{\mathrm{T}}$. Clearly, the vector process $\mathbf{g}_k$ in (34) depends not only on the time indexes but also on the timing offsets. Furthermore, we can show that (34) does not have its corresponding time-reversed process as given in (27). Consequently, the structure of $\mathbf{g}_k$ for the application of timing recovery considered in this work violates a GM process, thus preventing us from arriving at the exact expression of soft information as given in (29). This explains why, when developing the PSP-BCJR module based on the system model given in Figure 4, we cannot derive it analytically because the system model is too complicated. Instead, we attempt to develop PSP-BCJR in an *ad-hoc* manner by incorporating the timing update operation inside the BCJR algorithm so as to perform timing recovery and equalization jointly based on the BCJR algorithm.

It should also be noted that the PSP-BCJR module developed in this work can be applied to any application, and not limited to the channel without ISI as considered in [2, 3]. In the next section, we will explain how PSP-BCJR is developed and how it functions.

4.3 System Description

Consider the coded PR channel model in Figure 24. The message bits $\{x_k\}$ are encoded by an encoder and interleaved by an s-random interleaver[1] [30] (i.e., the π block) to form an interleaved sequence, a_k. The interleaved sequence a_k with bit

[1]It will randomly shuffle the symbols so that no two symbols within s symbols before the interleaver will be within s symbols afterwards.

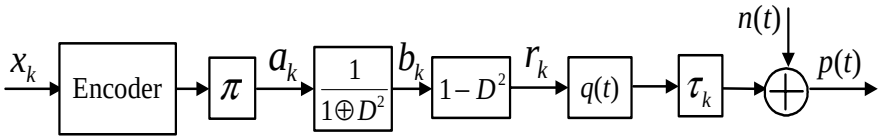


Figure 24: Data encoding with the precoded PR-IV channel model.

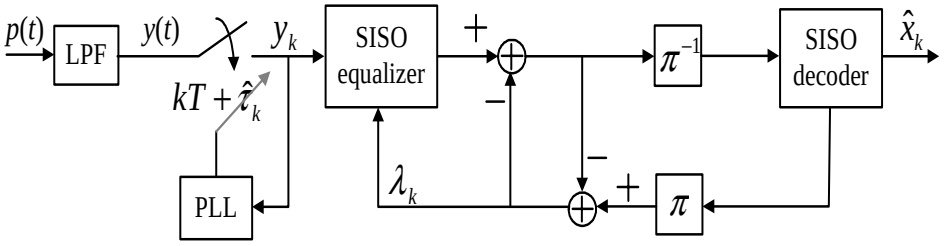


Figure 25: A conventional receiver architecture.

period T is further precoded by a $1/(1 \oplus D^2)$ precoder and modulated by a PR-IV pulse $h(t) = q(t) - q(t - 2T)$.

Again, we assume perfect acquisition by setting $\tau_0 = 0$. Because our model has no frequency offset component, the sampling phase offset can then be updated by a first-order PLL according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi \frac{3T}{16}\{y_k \tilde{r}_{k-1} - y_{k-1}\tilde{r}_k\}, \tag{35}$$

where y_k is the k-th sampler output (see Figure 25), and $\tilde{r}_k$ is the k-th *soft estimate* of the channel output, $r_k \in \{0, \pm 2\}$, which is given by [56]

$$\tilde{r}_k = E[r_k|y_k] = \frac{2\sinh(2y_k/\sigma_n^2)}{\cosh(2y_k/\sigma_n^2) + e^{2/\sigma_n^2}}. \tag{36}$$

The soft estimate is considered because it provides a better performance than the hard estimate [56] given in (17).

In the conventional receiver, conventional timing recovery is followed by a turbo equalizer [67, 75] as shown in Figure 25, which iteratively exchanges soft information between an SISO equalizer for the precoded PR-IV channel and an SISO decoder.

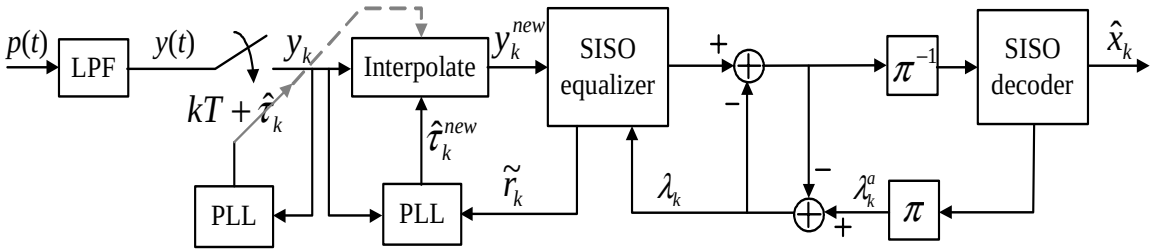


Figure 26: An NBM architecture.

4.4 Review of the NBM scheme

The NBM scheme was proposed by Nayak, Barry, and McLaughlin [56]. It is realized by embedding the timing recovery step inside the turbo equalizer so as to perform timing recovery, equalization, and error-correction decoding jointly. Figure 26 illustrates an architecture of the NBM scheme. We can summarize how the NBM scheme works as follows.

At each iteration, the turbo equalizer could produce the soft estimates $\{\tilde{r}_k\}$ that would be more reliable than the tentative decisions given in (36). For the precoded PR-IV channel, the soft estimate produced by the turbo equalizer is given by [56]

$$\tilde{r}_k = E[r_k|\{\lambda_k^a\}, \{\lambda_k^b\}] = \frac{-2\tanh(\lambda_{k-2}^b/2)}{1+e^{-\lambda_k^a}}, \tag{37}$$

where λ_k^a is the log-likelihood ratio (LLR) [8] of a_k produced by the SISO decoder, and λ_k^b is the LLR of b_k produced by the SISO equalizer.

By running the PLL again using the original readback waveform and the soft decision given in (37), we would get an improved set of sampling phase offsets $\{\hat{\tau}_k^{new}\}$. Note that, instead of storing the continuous-time readback signal, $y(t)$, we can store only the original set of samples, $\{y_k\}$, and its corresponding sampling phase offsets, $\{\hat{\tau}_k\}$, obtained from the first front-end PLL, because the bandlimited nature of $y(t)$ makes them sufficient statistics. Based on $\{y_k\}$, $\{\hat{\tau}_k\}$, and $\{\hat{\tau}_k^{new}\}$, an improved set of

samples, $\{y_k^{new}\}$, can be obtained by means of interpolation according to

$$y_k^{new} = \sum_i y_i q(kT + \hat{\tau}_k^{new} - iT - \hat{\tau}_i). \tag{38}$$

These new samples will then be used in the next iteration of the turbo equalizer. The process repeats as many iterations as needed. Clearly, the turbo equalizer benefits from better samples, and timing recovery benefits from better decisions.

In addition, Nayak, Barry, and McLaughlin also proposed a *simple* cycle slip detection and correction algorithm [56], which can be briefly explained as follows. As shown in Figure 10, a cycle slip leads to an abrupt change in $\hat{\tau}$ by $\pm T$. Then, a simple detection method is to declare a cycle slip whenever the magnitude of $\delta_k = \hat{\tau}_k$ - $\hat{\tau}_{k-d}$ exceeds a given threshold Δ_c, for some delay d. To correct a cycle slip, the detector needs to add $\pm T$ to all $\hat{\tau}$ after the cycle slip occurs, with the sign determined by the sign of δ_k. Unless otherwise stated, we use $\Delta_c = 0.75T$ and $d = 100T$ [56] in a cycle slip detection and correction algorithm for the NBM scheme throughout this work.

4.5 PSP-BCJR

As shown in (35), the performance of conventional timing recovery relies on the decision $\tilde{r}_k$ provided by its own symbol detector, which might yield an unreliable decision. To overcome this drawback, a reliable decision can be extracted by utilizing the already-given information inside the trellis structure [24]. Specifically, each state transition in the trellis uniquely specifies a corresponding symbol. Thus, at least one state transition in each trellis stage will correspond to a correct decision. Using that decision for the timing update operation will improve the performance of timing recovery. The idea of using the information available in the trellis to estimate other unknown parameters is known as PSP.

With PSP, we develop PSP-BCJR by embedding the timing recovery process inside the BCJR equalizer so as to perform timing recovery and equalization jointly. Figure 27 shows the PSP-BCJR algorithm, where the lines beginning with * are the

additional steps beyond the conventional BCJR algorithm. Note that the constant $3T/16$ in (B-9) and (B-22) is only for a PR-IV channel, and it can also be included in the PLL gain parameters.

Here, the key idea is that each node in the trellis has its own sampling phase offset. Thus, the branch metric is calculated based on the sampling phase offset of the starting state. Furthermore, we propose to update the timing estimate at each state based on the incoming branch that contributes the most to the state information. The detail on how PSP-BCJR performs during forward and backward recursions can be explained as follows.

4.5.1 Forward Recursion

Consider the PR-IV trellis structure shown in Figure 28. Let $\Psi_k = \{b_{k-1}\, b_{k-2}\}$ denote the state at time k. There are $Q = 2^\nu = 4$ states in this trellis, labeled as state 0 to state 3, where $\nu = 2$ is the precoded PR-IV channel memory. Let (p, q) be the *state transition* from state p to state q, and let $\pi_k(p)$ denote a *predecessor* for state p at time k, defined as the starting state associated with the *best* state transition leading to state p at time k. We define $\hat{\tau}_k(p)$ as the k-th *forward* sampling phase offset for state p at time k, which is used to sample $y(t)$ at time k for the state transition emanating from state p at time k, e.g., $y_k(p) = y(kT + \hat{\tau}_k(p))$, where $y_k(p)$ is the k-th sampler output for state p at time k. We also define $\hat{\theta}_k(p)$ as the k-th *forward* frequency error for state p at time k, which will be used to update $\hat{\tau}_k(p)$.

Consider the k-th stage of the trellis. There are two state transitions arriving at state 2 at time $k+1$, i.e., $(1, 2)$ and $(3, 2)$. First, we sample $y(t)$ using $\hat{\tau}_k(1)$ and $\hat{\tau}_k(3)$ to obtain $y_k(1)$ and $y_k(3)$, respectively. Next, we compute the metrics $\gamma_k(1, 2)$ and $\gamma_k(3, 2)$ based on (B-6), where $\hat{a}(p, q)$ is the interleaved bit (or the precoder input bit) associated with (p, q), and λ_k is the *a priori* LLR of a_k. Then, the state information $\alpha_{k+1}(2)$ is updated according to (B-7).

*(B-1) Initialize $\hat{\tau}_0(p) = 0$ and $\hat{\theta}_0(p) = 0$ for $\forall p$

(B-2) Initialize $[\alpha_0(0) \ldots \alpha_0(Q-1)] = [1\,0 \ldots 0]$

(B-3) For $k = 0, 1, \ldots, L+\nu-1$ [Forward recursion]

(B-4) For $q = 0, 1, \ldots, Q-1$

*(B-5) $y_k(p) = y(kT + \hat{\tau}_k(p))$ for $\forall p$

(B-6) $\gamma_k(p,q) = \exp\left\{-\frac{1}{2\sigma_n^2}|y_k(p) - \hat{r}(p,q)|^2 + \frac{\hat{a}(p,q)\lambda_k}{2}\right\}$ for $\forall p$

(B-7) $\alpha_{k+1}(q) = \sum_p \alpha_k(p)\gamma_k(p,q)$

*(B-8) $\pi_{k+1}(q) = \arg\max_p\{\alpha_k(p)\gamma_k(p,q)\}$

*(B-9) $\hat{\epsilon} = \frac{3T}{16}\{y_k(\pi_{k+1}(q))\hat{r}(\pi_k(\pi_{k+1}(q)), \pi_{k+1}(q)) - y_{k-1}(\pi_k(\pi_{k+1}(q)))\hat{r}(\pi_{k+1}(q), q)\}$

*(B-10) $\hat{\theta}_{k+1}(q) = \hat{\theta}_k(\pi_{k+1}(q)) + \kappa\hat{\epsilon}$

*(B-11) $\hat{\tau}_{k+1}(q) = \hat{\tau}_k(\pi_{k+1}(q)) + \xi\hat{\epsilon} + \hat{\theta}_{k+1}(q)$

(B-12) End

(B-13) End

*(B-14) Initialize $\hat{\tau}^b_{L+\nu}(p) = \hat{\tau}_{L+\nu}(p)$ and $\hat{\theta}^b_{L+\nu}(p) = \hat{\theta}_{L+\nu}(p)$ for $\forall p$

(B-15) $[\beta_{L+\nu}(0) \ldots \beta_{L+\nu}(Q-1)] = [1\,0 \ldots 0]$

(B-16) For $k = L+\nu-1, L+\nu-2, \ldots, 0$ [Backward recursion]

(B-17) For $p = 0, 1, \ldots, Q-1$

*(B-18) $y^b_k(q) = y(kT + \hat{\tau}^b_{k+1}(q))$ for $\forall q$

(B-19) $\gamma^b_k(p,q) = \exp\left\{-\frac{1}{2\sigma_n^2}|y^b_k(q) - \hat{r}(p,q)|^2 + \frac{\hat{a}(p,q)\lambda_k}{2}\right\}$ for $\forall q$

(B-20) $\beta_k(p) = \sum_q \gamma^b_k(p,q)\beta_{k+1}(q)$

*(B-21) $\pi^b_k(p) = \arg\max_q\{\gamma^b_k(p,q)\beta_{k+1}(q)\}$

*(B-22) $\hat{\epsilon} = \frac{3T}{16}\{y^b_k(\pi^b_k(p))\hat{r}(\pi^b_k(p), \pi^b_{k+1}(\pi^b_k(p))) - y^b_{k+1}(\pi^b_{k+1}(\pi^b_k(p)))\hat{r}(p, \pi^b_k(p))\}$

*(B-23) $\hat{\theta}^b_k(p) = \hat{\theta}^b_{k+1}(\pi^b_k(p)) + \kappa\hat{\epsilon}$

*(B-24) $\hat{\tau}^b_k(p) = \hat{\tau}^b_{k+1}(\pi^b_k(p)) + \xi\hat{\epsilon} + \hat{\theta}^b_k(p)$

*(B-25) $\hat{\tau}^b_k(p) = \{\hat{\tau}^b_k(p) + \hat{\tau}_k(p)\}/2$ if $|\hat{\tau}^b_k(p) - \hat{\tau}_k(p)| > \Delta$

(B-26) End

(B-27) $\lambda^p_k = \log\left\{\frac{\sum_{\{p,q:\ \hat{a}(p,q)=+1\}} \alpha_k(p)\gamma^b_k(p,q)\beta_{k+1}(q)}{\sum_{\{p,q:\ \hat{a}(p,q)=-1\}} \alpha_k(p)\gamma^b_k(p,q)\beta_{k+1}(q)}\right\}$

(B-28) End

Figure 27: PSP-BCJR algorithm, where the lines beginning with * are the additional steps beyond the conventional BCJR algorithm.

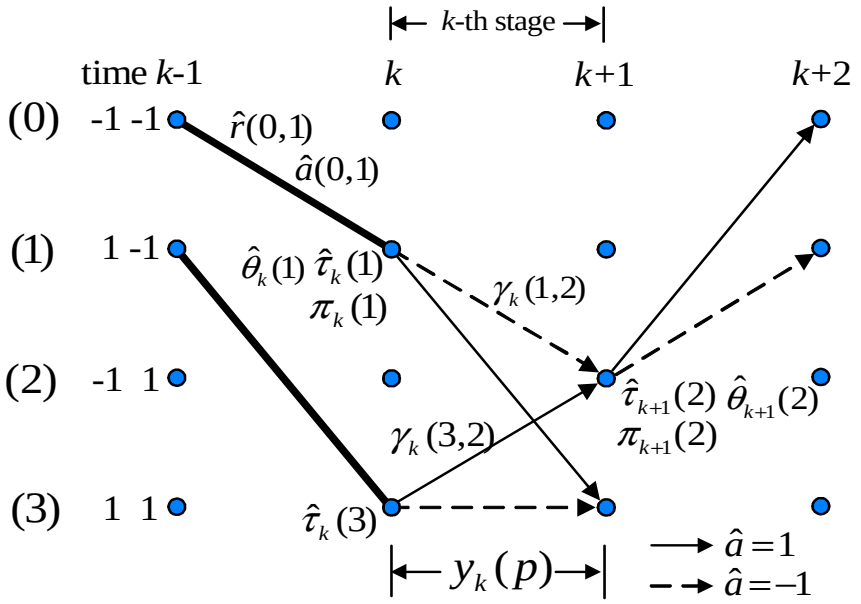


Figure 28: The PR-IV trellis structure demonstrating how PSP-BCJR performs during forward recursion.

The starting state associated with the best state transition leading to state 2 at time $k+1$ is chosen according to (B-8). Suppose $(1,2)$ is the best state transition leading to state 2 at time $k+1$ so that $\pi_{k+1}(2) = 1$. We update the next forward sampling phase offset, $\hat{\tau}_{k+1}(2)$, based on (B-9) – (B-11). This $\hat{\tau}_{k+1}(2)$ will be used to sample $y(t)$ at time $k+1$ for the state transitions emanating from state 2 at time $k+1$.

4.5.2 Backward Recursion

The backward timing update operation serves as refining the samples $\{y_k\}$ so as to improve the quality of the branch metrics. To explain how it works, we introduce the *backward transition* represented by the gray arrows as shown in Figure 29. Let $\pi^b_{k+1}(q)$ be a *successor* for state q at time $k+1$, defined as the starting state associated with the *best* backward transition leading to state q at time $k+1$. We define $\hat{\tau}^b_{k+1}(q)$ as the k-th *backward* sampling phase offset for state q at time $k+1$, which is used to sample $y(t)$ at time k during backward recursion, e.g., $y^b_k(q) = y(kT + \hat{\tau}^b_{k+1}(q))$.

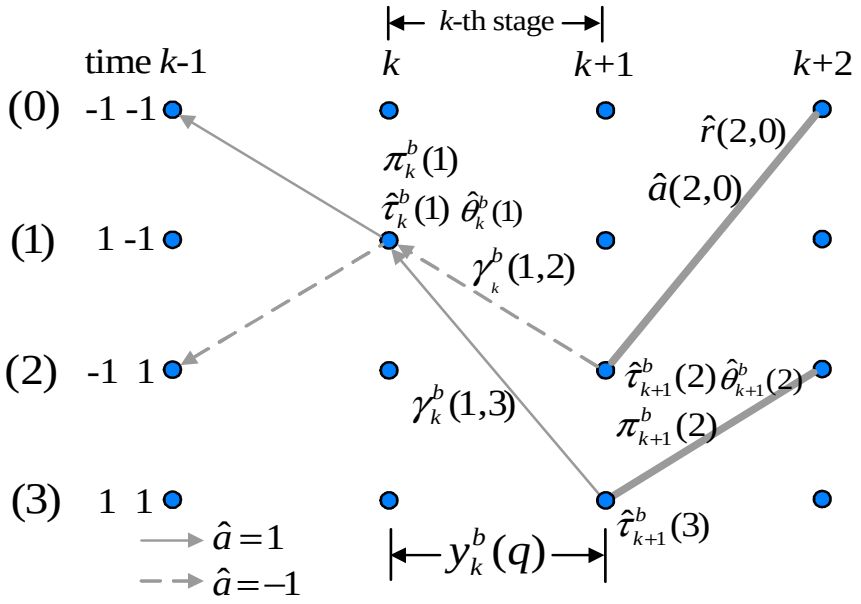


Figure 29: The PR-IV trellis structure illustrating how PSP-BCJR performs during backward recursion.

We also define $\hat{\theta}_{k+1}^b(q)$ as the k-th *backward* frequency error for state q at time $k+1$, which will be used to update $\hat{\tau}_{k+1}^b(q)$.

Consider the backward transition at the k-th stage. There are two backward transitions arriving at state 1 at time k, which correspond to $(1,2)$ and $(1,3)$. We first sample $y(t)$ using $\hat{\tau}_{k+1}^b(2)$ and $\hat{\tau}_{k+1}^b(3)$ to obtain $y_k^b(2)$ and $y_k^b(3)$, respectively. Then, we compute the metrics $\gamma_k^b(1,2)$ and $\gamma_k^b(1,3)$ based on (B-19), and update the state information $\beta_k(1)$ using (B-20). Similarly, the starting state associated with the best backward transition leading to state 1 at time k is selected according to (B-21). Suppose $(1,2)$ corresponds to the best backward transition leading to state 1 at time k so that $\pi_k^b(1) = 2$. The next backward sampling phase offset, $\hat{\tau}_k^b(1)$, is updated by (B-22) – (B-24).

To avoid a cycle slip when $\hat{\tau}_k^b(1)$ starts deviating from $\hat{\tau}_k(1)$, we propose a simple remedy by averaging the backward sampling phase offset according to (B-25), where Δ is the threshold that allows the backward sampling phase offset to deviate from the

forward one. In this work, we set $\Delta = 0.1T$ because we want to keep $\{\hat{\tau}_k^b\}$ close to $\{\hat{\tau}_k\}$. This $\hat{\tau}_k^b(1)$ will be used to sample $y(t)$ at time $k-1$ for the backward transitions emanating from state 1 at time k. Finally, the *a posteriori* LLR of a_k, λ_k^p, is computed based on (B-27).

4.5.3 Complexity of PSP-BCJR

It is apparent that, for the precoded PR-IV channel, PSP-BCJR requires eight PLLs, i.e., one PLL for each state in one stage of the trellis during both forward and backward recursions. Furthermore, instead of storing the received analog signal $y(t)$, we could uniformly sample $y(t)$ at symbol rate to obtain a set of samples $\{y_k\}$. Then, we can store only this set of samples because the bandlimited nature of $y(t)$ makes it sufficient statistics. Consequently, PSP-BCJR can perform the timing update operation using $\{y_k\}$ and a digital interpolation filter, thus decreasing its complexity. Unless otherwise specified, in this work, a 21-tap sinc interpolation filter is employed.

Beyond the conventional BCJR algorithm, PSP-BCJR needs new storage requirements for (i) the forward/backward sampling phase offsets, (ii) the forward/backward frequency errors, (iii) the starting states, and (iv) the sampler outputs. However, there is no need to store the whole sets of the backward sampling phase offsets, the starting states, and the sampler outputs. In other words, only those of the *current* and *previous* stages need to be stored, thus minimizing the need for extra memory.

To help quantify how much computational complexity PSP-BCJR contains, we measure its complexity by counting the total number of additions and multiplications. For other mathematical functions, such as $\log(x)$, $\exp(x)$, etc., we assume that they can be implemented as lookup tables, and that we ignore their complexity.

Table 1 shows the complexity of each module that is employed in iterative timing recovery schemes, where N_{sinc} is the total number of taps of the sinc interpolation filter. The N_{sinc}-tap sinc interpolation filter is used to refine the samples for each

Table 1: The total number of operations of each module that is used in iterative timing recovery schemes.

Module	Number of operations	
	Addition	Multiplication
Sinc interpolation (per sample)	$4N_{sinc} - 1$	N_{sinc}
PLL with hard decision (per sample)	2	3
PLL with soft decision (per sample)	3	6
BCJR equalizer (per bit)	$12Q - 2$	$20Q + 1$
PSP-BCJR (per bit)	$(14 + 8N_{sinc})Q - 2$	$(26 + 2N_{sinc})Q + 1$

iteration according to (38), where the running indexes k and i are normalized with respect to T. The number of operations of PLL is counted based on (18) for hard decision, and (35) – (36) for soft decision. Note that PSP-BCJR performs interpolation and the timing update operation at each state during both forward and backward recursions. As shown in Table 1, it is obvious that PSP-BCJR has higher complexity than the conventional BCJR algorithm, especially when N_{sinc} is large. This is due to the timing update operation performed at each trellis state during both forward and backward recursions.

4.6 Numerical Results and Discussion

The per-survivor iterative timing recovery scheme is easily obtained by discarding the front-end PLL in Figure 25 and replacing the BCJR equalizer with PSP-BCJR.

Consider a rate-8/9 system in which a block of 3636 message bits is encoded by a rate-1/2 recursive systematic convolutional (RSC) encoder with a generator polynomial $[1, \frac{1 \oplus D \oplus D^3 \oplus D^4}{1 \oplus D \oplus D^4}]$, and is then punctured to a block length of 4095 bits by retaining only every the eighth parity bit. The punctured sequence passes through an s-random interleaver with $s = 16$ to obtain an interleaved sequence of a_k. Both the SISO equalizer and the SISO decoder are implemented based on a BCJR algorithm. Again, the PLL gain parameters for different iterative timing recovery schemes were

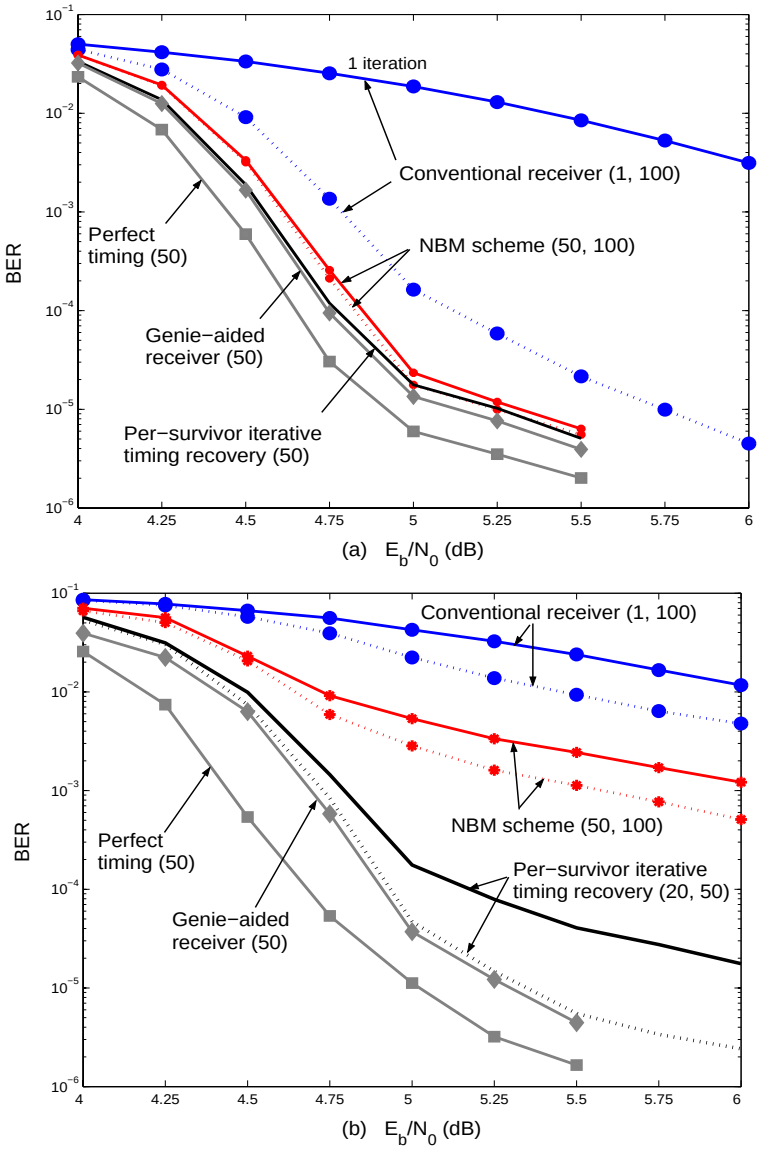


Figure 30: Performance comparison of different iterative timing recovery schemes for (a) $\sigma_w/T = 0.5\%$ and (b) $\sigma_w/T = 1\%$.

optimized based on minimizing σ_ϵ/T at $E_b/N_0 = 5$ dB. Each BER point was computed using as many data sectors as possible until at least 100 sectors in error were collected at the 100-th iteration.

Figure 30(a) compares the BER performance of different iterative timing recovery schemes with $\sigma_w/T = 0.5\%$, which implies a low probability of occurrence of a cycle slip. Note that the number inside the parenthesis in Figure 30 indicates the total

number of iterations used to generate each curve. The curve labeled "Perfect timing" represents the conventional receiver that uses $\hat{\tau}_k = \tau_k$ to sample $y(t)$. Furthermore, the curve labeled "Genie-aided receiver" represents the conventional receiver whose PLL has access to all correct decisions, thus serving as a lower bound for a timing recovery scheme that is based on a PLL. The ξ's for the conventional receiver, the NBM scheme, per-survivor iterative timing recovery, and the genie-aided receiver are 0.0053, 0.0053, 0.0028, and 0.0036, respectively. As depicted in Figure 30(a), per-survivor iterative timing recovery performs slightly better than the NBM scheme at the 50-th iteration, and both yield about a 0.45 dB gain at BER = 10^{-5} over the conventional receiver. Note that the performances of the conventional receiver at the 50-th and the 100-th iterations are alike (not shown). In addition, per-survivor iterative timing recovery performs nearly as well as the genie-aided receiver and is only a 0.35 dB away from the system with perfect timing at BER = 10^{-5}.

Next, let us consider the system with a severe random walk parameter $\sigma_w/T = 1\%$, which implies a high probability of occurrence of a cycle slip. The ξ's for the conventional receiver, the NBM scheme, per-survivor iterative timing recovery, and the genie-aided receiver are 0.0103, 0.0103, 0.006, and 0.007, respectively. Figure 30(b) shows the BER performance of different iterative timing recovery schemes with $\sigma_w/T = 1\%$. The NBM scheme still outperforms the conventional receiver; however, it seems to have an error floor at high BER. On the other hand, per-survivor iterative timing recovery yields a large performance gain over the NBM scheme and starts to have an error floor at low BER. Again, per-survivor iterative timing recovery still performs close to the genie-aided receiver and loses approximately a 0.35 dB relative to the system with perfect timing at BER = 10^{-5}.

The reason that per-survivor iterative timing recovery outperforms the NBM scheme when σ_w/T is large might be because the front-end PLL used in the NBM scheme does not work well if compared to PSP-based timing recovery, as studied in

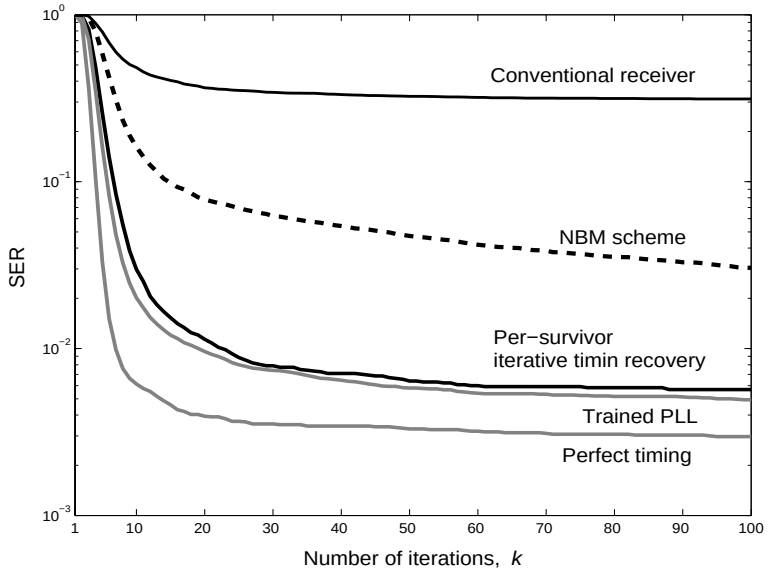


Figure 31: Convergence rate of different iterative timing recovery schemes at E_b/N_0 = 5 dB and $\sigma_w/T = 1\%$.

Section 3. Additionally, we observed that per-survivor iterative timing recovery can automatically and rapidly correct a cycle slip (without a cycle slip detection and correction technique as employed in the NBM scheme [56]) much more efficiently than the NBM scheme. In other words, per-survivor iterative timing recovery can achieve faster convergence than the NBM scheme, which can be confirmed by plotting the sector-error rate (SER) versus the number of iterations in Figure 31. It is evident that the convergence rate of per-survivor iterative timing recovery is very close to that of the genie-aided receiver, which takes about 30 iterations to provide a good performance. Conversely, the NBM scheme takes hundreds of iterations to yield a good performance (not shown).

We also plot the probability of an uncorrected cycle slip in Figure 32, where we declare a cycle slip when the actual timing offset and the estimated one are $0.75T$ apart from each other for more than 100 consecutive bit periods. This plot will show how fast each scheme can correct a cycle slip. Apparently, the NBM scheme requires a large number of iterations to correct a cycle slip as opposed to per-survivor iterative

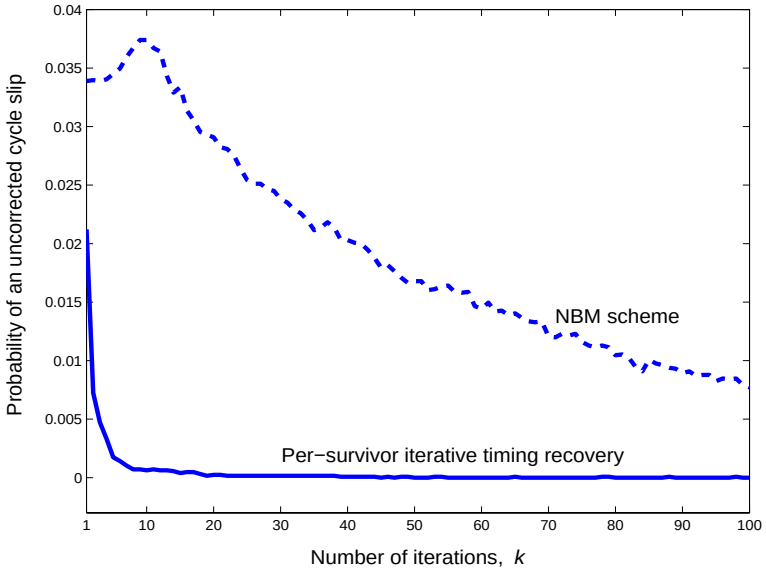


Figure 32: Probability of an uncorrected cycle slip at $E_b/N_0 = 5$ dB and $\sigma_w/T = 1\%$.

timing recovery. The reason for this might be because the NBM scheme can correct a cycle slip only when there is a *sudden* phase change in the estimated timing offsets, not in the actual timing offsets. It should be noted that the reason that the NBM scheme increases the probability of an uncorrected cycle slip at the first 10 iterations (see Figure 32) might possibly be because the NBM scheme takes a few iterations for the cycle slip detection algorithm to recognize a cycle slip according to our cycle slip criterion.

It is also worth plotting the estimated timing offset obtained from the NBM scheme and per-survivor iterative timing recovery for two different sample packets as depicted in Figure 33. As illustrated in Figure 33(a), the NBM scheme takes about 150 iterations to correct a cycle slip (not shown), whereas per-survivor iterative timing recovery takes only one iteration. This implies that the NBM scheme corrects a cycle slip slowly. Similarly, Figure 33(b) indicates that per-survivor iterative timing recovery can correct a cycle slip within 5 iterations but the NBM scheme cannot correct a cycle slip even with 50 iterations. This suggests that per-survivor iterative timing recovery can correct a cycle slip quickly. In conclusion, per-survivor iterative timing

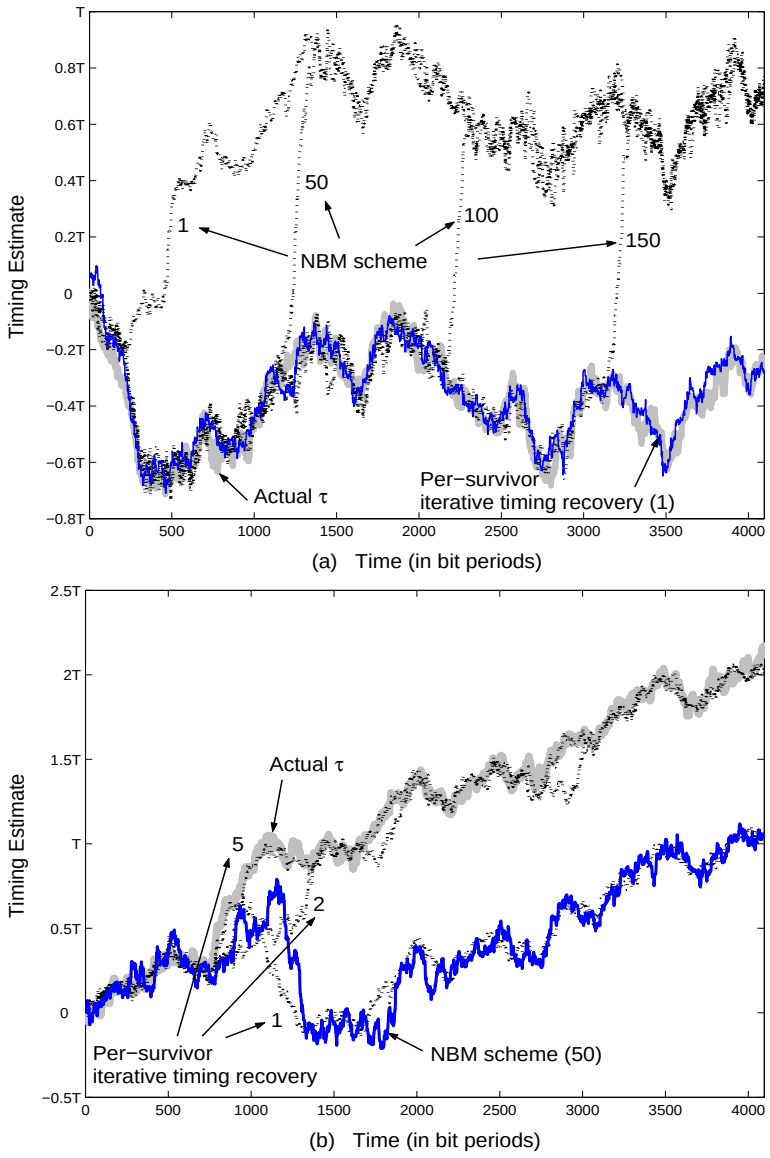


Figure 33: Cycle slip correction for two different sample packets at $E_b/N_0 = 5$ dB and $\sigma_w/T = 1\%$.

recovery is a good scheme to employ in a system that experiences a high probability of occurrence of a cycle slip.

4.6.1 Complexity Versus Performance

To compare the complexity of different iterative timing recovery schemes, we include the complexity of the SISO decoder. Let χ be code memory of a convolutional code,

Table 2: The total number of operations of each iterative timing recovery scheme for a coded PR-IV channel.

Schemes	Number of operations (per bit)		
	Addition	Multiplication	Total
Conventional receiver	$86 + 330N$	$27 + 595N$	$113 + 925N$
NBM scheme	$416N$	$622N$	$1038N$
Per-survivor iterative	$1010N$	$787N$	$1797N$
Genie-aided receiver	$85 + 330N$	$24 + 595N$	$109 + 925N$
Perfect timing	$83 + 330N$	$21 + 595N$	$104 + 925N$

which is equal to 4 (see the generator polynomial given in Section 4.6). Thus, the SISO decoder implemented based on the BCJR algorithm will have $S = 2^{\chi} = 16$ states in one trellis stage. It can be shown that the BCJR decoder requires $18S - 4$ additions and $32S + 2$ multiplications. Based on Table 1, we can then count the total number of operations of each iterative timing recovery scheme for $Q = 4$, $N_{sinc} = 21$, and $S = 16$, as illustrated in Table 2, where N is the number of iterations. Assuming that addition and multiplication have the same complexity, Figure 34 compares the total number of operations of each scheme. Clearly, per-survivor iterative timing recovery has much higher complexity than the others, and the NBM scheme has complexity comparable to the conventional receiver (as stated in [56]).

It is worth comparing the performance of different iterative timing recovery schemes when they approximately have the same complexity. To do so, we first assume that the current technology can support the total number of operations equal to 5 iterations of per-survivor iterative timing recovery. As shown in Table 2, it can be shown that per-survivor iterative timing recovery with 5 iterations has the total number of operations approximately equal to the conventional receiver with 10 iterations, the NBM scheme with 9 iterations, the genie-aided receiver with 10 iterations, and the system with perfect timing with 10 iterations.

Figure 35 compares the performance of different iterative timing recovery schemes

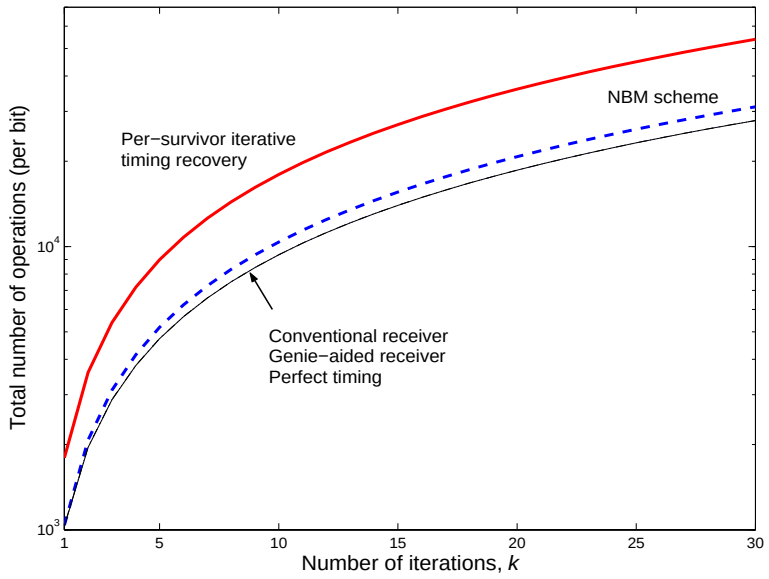


Figure 34: Complexity comparison of different iterative timing recovery schemes for $N_{sinc} = 21$, $Q = 4$, and $S = 16$.

when they approximately have the same complexity for systems with $\sigma_w/T = 0.5\%$ and $\sigma_w/T = 1\%$. As shown in Figure 35(a), per-survivor iterative timing recovery still performs better than the conventional receiver and it performs as well as the NBM scheme. This suggests that there is no need to use per-survivor iterative timing recovery when a channel experiences a low probability of occurrence of a cycle slip. A simple iterative timing recovery scheme like the NBM scheme can provide a good performance with complexity comparable to the conventional receiver. On the other hand, per-survivor iterative timing recovery can provide a large performance gain over the NBM scheme when operating in a system with $\sigma_w/T = 1\%$, as depicted in Figure 35(b). This gain comes from the fact that per-survivor iterative timing recovery can correct a cycle slip rapidly, as opposed to the NBM scheme. Therefore, it can be concluded that, for low to moderate complexity, per-survivor iterative timing recovery still performs better than conventional schemes, especially when operating in a system that experiences a high probability of occurrence of a cycle slip (e.g., a system with large timing errors).

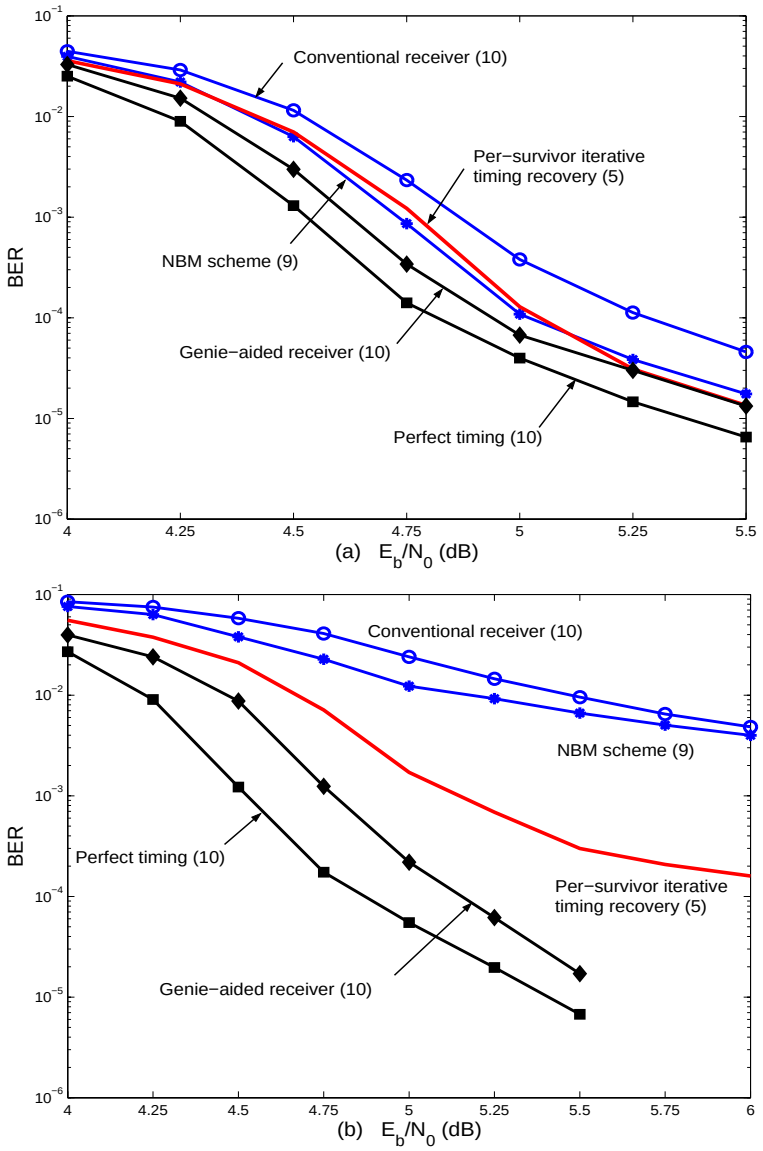


Figure 35: BER performance of different iterative timing recovery schemes when they approximately have the same complexity for system with (a) $\sigma_w/T = 0.5\%$ and (b) $\sigma_w/T = 1\%$.

Note that the results presented in this section are based on a system whose ECC is a convolutional code. Thus, the SISO decoder implemented based on the BCJR algorithm has very high complexity (i.e., it requires 284 additions and 514 multiplications per bit). Although other ECCs (e.g., a low-density parity-check (LDPC) code [26]) might be used to reduce the complexity of the SISO decoder, similar results are

still obtained. Consequently, we would like to point out that conclusion drawn from this section is still valid for any ECC.

4.7 Reduced-Complexity PSP-BCJR

Because the BCJR algorithm performs on the same trellis as the Viterbi algorithm does, many approaches have been proposed [16, 25, 78] to reduce its complexity, including the M- and T- algorithms [25] (similar to the ones presented in Section 3.6). In this section, however, we again focus only on the M- and T- algorithms to reduce the complexity of PSP-BCJR because of their simplicity.

4.7.1 The M-algorithm

At each time instant, the M-algorithm first finds the *maximum* path metric leading to each trellis state. Note that we can consider the state information (i.e., $\alpha_k(p)$ or $\beta_k(p)$) at each state as the path metric. Then, it retains only the M (M must be less than the total number of states in one stage of the trellis) paths with the *highest* path metrics among all paths; the rest are declared *dead* and their corresponding state information are set to zero. The same procedure is also applied to the backward recursion. Because the LLR output, λ_k^p, (see (B-27) in Figure 27) is the products of the state information, it is then simpler to operate the backward recursion only on the region of the trellis where the forward components are *alive*.

4.7.2 The T-algorithm

Again, we can consider the state information (i.e., $\alpha_k(p)$ or $\beta_k(p)$) at each state as the path metric, and we must normalize the state information at each time instant so that the summation of all state information is one, i.e., $\sum_p \alpha_k(p) = 1$ and $\sum_p \beta_k(p) = 1$. Then, the T algorithm discards all paths whose path metrics fall below a threshold T; the rest are declared *dead* and their corresponding state information are set to zero. Similarly, the backward recursion can only operate through the region where

the forward components are *alive*.

4.7.3 Performance Comparison

To compare the performance of per-survivor iterative timing recovery with different reduced-complexity approaches, we again consider the SER and the average number of searched states, as used in Section 3.6.3.

Figure 36 compares the performance of per-survivor iterative timing recovery with different reduced-complexity approaches at the 10-th iteration with $\sigma_w/T = 0.5\%$. As expected, there is a trade-off between the SER and the average number of searched states. In other words, the larger the average number of searched states, the lower the SER. At low E_b/N_0, the M-algorithm does not perform well (same reasons as explained in Section 3.6.3). For this specific channel model, although the M-algorithm with M = 3 and the T-algorithm with T = 0.00001 are comparable in terms of SER at $E_b/N_0 \approx 5$ dB, the T-algorithm is preferred because it requires a fewer number of searched states. We also observed that a similar result is obtained for any σ_w/T.

4.7.4 Note on Complexity Reduction of PSP-BCJR

We can even further reduce the complexity of PSP-BCJR with negligible performance loss, which can be explained as follows.

- As shown in Table 1, the complexity of PSP-BCJR partly depends on N_{sinc}. A small N_{sinc} can be used to reduce the complexity of PSP-BCJR at the expense of some performance loss. Alternatively, we can utilize other interpolation filters that have a fewer number of taps but still provide a good performance, if compared against a 21-tap sinc interpolation filter. A minimum mean-squared error (MMSE) interpolation filter [87] can be given as an example, which can yield a good performance with only 8 taps.

- We have observed through simulation that the sampling phase offset associated

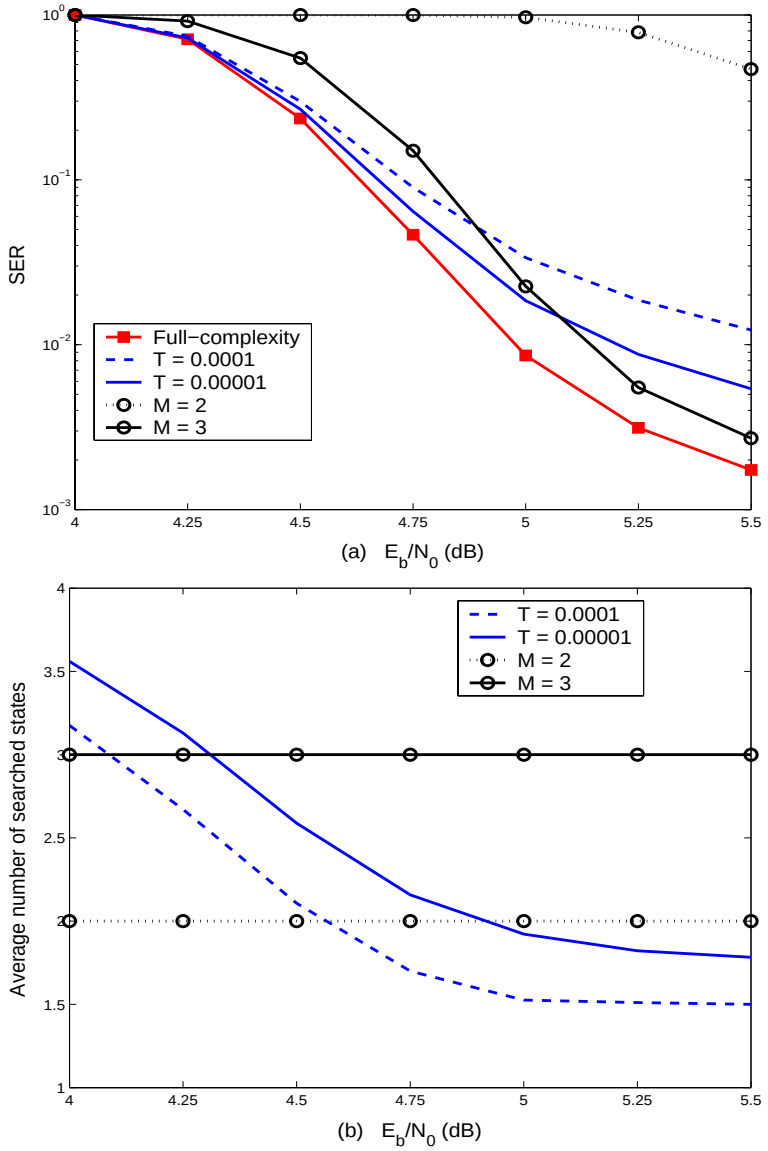


Figure 36: Performance comparison of per-survivor iterative timing recovery with different reduced-complexity approaches at the 10-th iteration with $\sigma_w/T = 0.5\%$.

with each state in one trellis stage is close to one another. Therefore, it might be possible to utilize only those with the highest and the lowest values, say $\hat{\tau}_k^{\max}$ and $\hat{\tau}_k^{\min}$, to obtain the sampler outputs, say $y_k^{\max}$ and $y_k^{\min}$. Then, the other sampler outputs associated with the other sampling phase offsets can be obtained by means of linear interpolation, based on $\hat{\tau}_k^{\max}$, $\hat{\tau}_k^{\min}$, $y_k^{\max}$, and $y_k^{\min}$. This will help reduce the complexity of PSP-BCJR considerably.

4.8 Extrinsic Information Transfer (EXIT) Chart

Performance analysis of iterative timing recovery schemes is difficult because of their complexity. Accordingly, a time-consuming simulation in terms of BER is practically a solution to compare their performances as presented in Section 4.6.

The *extrinsic information transfer chart* (EXIT chart) was proposed by ten Brink [80] as a tool for predicting the convergence behavior of turbo codes [10, 30]. The key idea is that the SISO equalizer and the SISO decoder in an iterative receiver can be modeled as devices mapping the extrinsic information from its input to its output. Because the output of one SISO module is an input to the other SISO module and vice versa, these two transfer characteristics can be plotted in a single diagram with the axes of the decoder transfer characteristic swapped. Furthermore, the exchange of extrinsic information can be visualized as a *system trajectory* between these two transfer characteristics, which can be accurately used to predict the performance of iterative decoding schemes without simulating data transmission on the complete iterative receiver [80].

Recently, the EXIT chart has been employed to analyze the performance of turbo equalization as studied in [79] for the case of a known, time-invariant channel, and in [61] for the adaptive turbo equalization, both assuming that the synchronization is perfect. In this section, we will demonstrate how the EXIT chart can also be used to predict the performance of the system with imperfect synchronization or, specifically, the iterative timing recovery scheme. We also validate the use of the EXIT chart instead of BER as a convenient measure to compare the performance of different iterative timing recovery schemes, considering that the BER computation takes a considerable amount of simulation time.

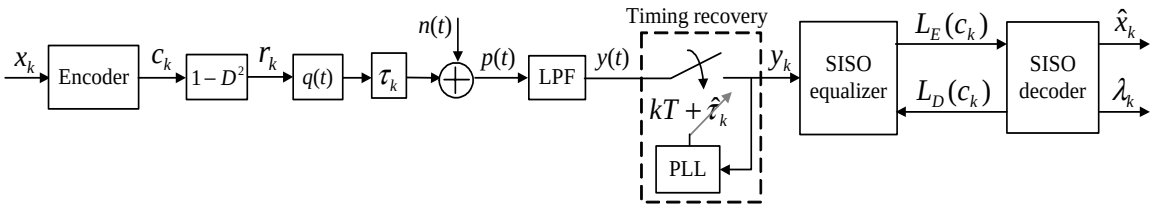


Figure 37: A system model for the EXIT chart analysis.

4.8.1 EXIT Chart for Iterative Timing Recovery

To simplify our discussion, we consider the system model shown in Figure 37 and assume perfect acquisition. Because our model has no frequency offset component, the sampling phase offset can then be updated by a first-order PLL as given in (35).

The SISO equalizer takes the channel observations $\{y_k\}$ and the *a priori* information $L_D(c_k)$ to output the extrinsic information $L_E(c_k)$, which becomes the *a priori* input for the SISO decoder. The SISO decoder outputs soft values $\{\lambda_k\}$ and feeds back the extrinsic information $L_D(c_k) = \lambda_k - L_E(c_k)$ to become the *a priori* input of the SISO equalizer. Note that the variables L and λ are LLRs.

The analysis tool called the EXIT chart is used to graphically describe the convergence behavior of the iterative decoding scheme by investigating the exchange of *mutual information* between the equalizer and the decoder. To obtain the EXIT chart, the mutual information at the equalizer output, $I_E^{out} = I(L_E(c_k); c_k)$, and at the decoder output, $I_D^{out} = I(L_D(c_k); c_k)$, is needed, where the mutual information is defined as [80]

$$I(L;C) = \frac{1}{2} \sum_{c\in\{\pm 1\}} \int_{-\infty}^{\infty} p(l|c) \cdot \log_2 \left(\frac{2p(l|c)}{p(l|1) + p(l|-1)} \right) dl, \tag{39}$$

where $p(l|c) = p(l|C = c)$ denotes a probability density function (pdf) of the extrinsic information L given that c was transmitted. The range of $I(L;C)$ is $[0, 1]$, where $I(L;C) = 0$ or $I(L;C) = 1$ means no or perfect knowledge of the transmitted bit c. Equation (39) is evaluated by first calculating the histograms of $c_k L(c_k)$ so as to

estimate $p(l|1)$ and $p(l|-1)$. Hence, $I(L;C)$ is obtained by numerically computing the integral in (39).

4.8.2 Simulation Setup

Figure 38 shows a simulation setup for generating the mutual information transfer characteristics of the conventional receiver. The *a priori* information is usually modeled as a normal distribution with average value $c_k\sigma_A^2/2$ and variance σ_A^2 [80]. The mutual information at the input of the SISO module is evaluated using (39), which, for an *i.i.d.* binary sequence, reduces to [80]

$$I_A(\sigma_A) = 1 - \int_{\infty}^{\infty} \frac{e^{-(l-\sigma_A^2/2)^2/(2\sigma_A^2)}}{\sqrt{2\pi}\sigma_A} \cdot \log_2(1+e^{-l})dl. \tag{40}$$

Clearly, I_A is independent of the corresponding transmitted bits. Thus, it can be precomputed and tabulated.

To obtain the decoder transfer characteristic, a simulation setup shown in Figure 38(a) is used. By varying σ_A^2 so that I_A ranges from 0 to 1, the decoder transfer characteristic (a plot between I_A and I_D^{out}) is obtained. Similarly, the equalizer transfer characteristic (a plot between I_A and I_E^{out}) of the conventional receiver can be obtained by running a simulation in Figure 38(b), where the data bits $\{c_k\}$ are randomly generated without the encoder. Then, the EXIT chart is realized by combining the transfer characteristics of the equalizer and the decoder into a single diagram where the axes of the decoder are swapped. The exchange of the mutual information, I_E^{out} and I_D^{out}, over the iterations in the complete receiver can be visualized as a system trajectory in the EXIT chart [80].

To obtain the equalizer transfer characteristic of the NBM scheme, the simulation setup in Figure 38(b) needs to be modified as illustrated in Figure 39, where $\lambda_k = L_D(c_k) + L'_E(c_k)$ is the LLR at the output of the SISO decoder. Given $\{\lambda_k\}$, the soft estimates $\{\tilde{r}_k\}$ for the PR-IV channel can be expressed as [34]

$$\tilde{r}_k = E[r_k|\{\lambda_k\}] = \frac{2(e^{\lambda_k} - e^{\lambda_{k-2}})}{(1+e^{\lambda_k})(1+e^{\lambda_{k-2}})}. \tag{41}$$

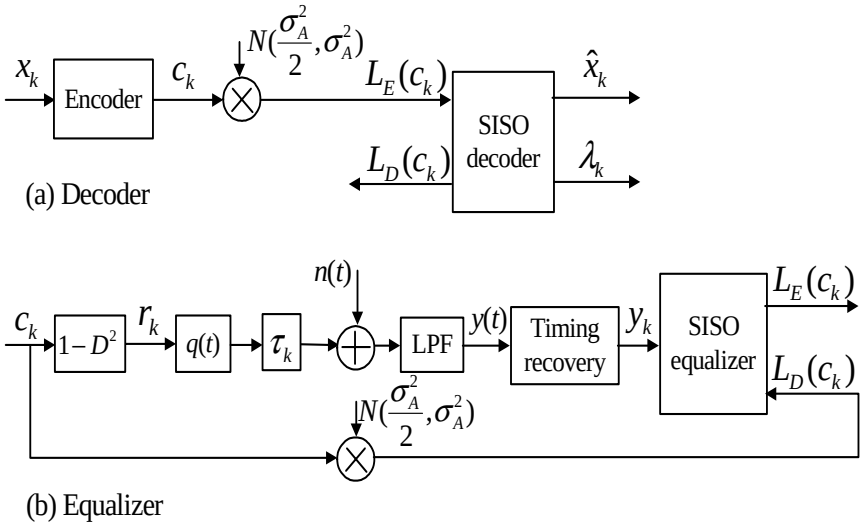


Figure 38: Simulation setup for generating the mutual information transfer characteristics of the conventional receiver for (a) the decoder and (b) the equalizer.

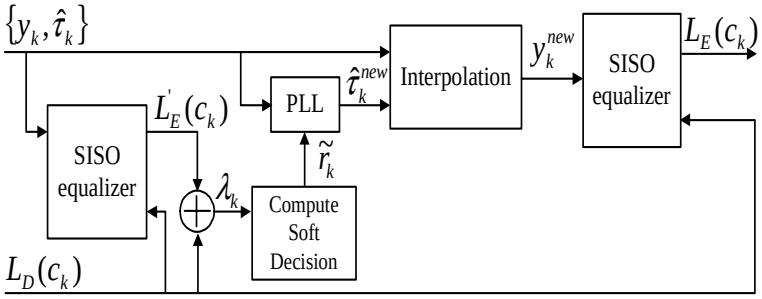


Figure 39: Simulation setup for generating the equalizer transfer characteristic of the NBM scheme.

The second PLL uses these soft estimates and the original samples $\{y_k\}$ to output an improved set of the sampling phase offsets, $\{\hat{\tau}_k^{new}\}$, which will be used to resample $\{y_k\}$ by means of interpolation according to (38). A new set of the samples, $\{y_k^{new}\}$, is then used to compute the equalizer transfer characteristic.

Finally, for the equalizer transfer characteristic of per-survivor iterative timing recovery, we only replace the timing recovery block and the SISO equalizer in Figure 38(b) with a PSP-BCJR module and then run a simulation to generate the equalizer transfer characteristic as described earlier.

4.8.3 Predicted Bit-Error Rate

As shown in (40), there is one-to-one mapping between σ_A^2 and I_A. Thus, if we define [80]

$$J(\sigma) := I_A(\sigma_A = \sigma) \tag{42}$$

with

$$\lim_{\sigma \to 0} J(\sigma) = 0 \text{ and } \lim_{\sigma \to \infty} J(\sigma) = 1, \text{ for } \sigma > 0, \tag{43}$$

it is true that

$$\sigma_A = J^{-1}(I_A) \tag{44}$$

because I_A is monotonically increasing in σ_A and thus reversible [80]. It should be noted that the relationship in (44) is also valid for the mutual information at the output of both the equalizer and the decoder [80].

Consequently, we can predict the BER performance evaluated at the SISO decoder output, whose LLR is given by $\lambda_k = L_E(c_k) + L_D(c_k)$, as follows. Assuming that λ and L are Gaussian distributed and independent, we can write

$$\sigma_\lambda^2 = \sigma_E^2 + \sigma_D^2, \tag{45}$$

where

$$\sigma_E^2 \approx (J^{-1}(I_D^{in}))^2, \tag{46}$$

$$\sigma_D^2 \approx (J^{-1}(I_D^{out}))^2, \tag{47}$$

and $I_D^{in} = I_E^{out}$.

For simplicity, we also assume that $\lambda \sim \mathcal{N}(\sigma_\lambda^2/2, \sigma_\lambda^2)$. With the complementary error function [8], the error probability at the output of the SISO decoder, P_e, can be approximately expressed as [34]

$$\begin{aligned} P_e &\approx \frac{1}{2}\operatorname{erfc}\left(\frac{\sigma_\lambda}{2\sqrt{2}}\right) \\ &\approx \frac{1}{2}\operatorname{erfc}\left(\frac{\sqrt{(J^{-1}(I_D^{in}))^2 + (J^{-1}(I_D^{out}))^2}}{2\sqrt{2}}\right). \end{aligned} \tag{48}$$

4.8.4 Performance Comparison

Consider a rate-8/9 system in which a block of 3640 message bits is encoded by a regular $(3, 27)$ LDPC code [26], resulting in a coded block length of 4095 bits. The parity-check matrix has 3 ones in each column and 27 ones in each row. With an LDPC outer code, the interleaver is not needed. Note that an LDPC code is considered in this section instead of a convolution code (as used in Section 4.6) because we want to show that all results presented in this work are also valid for any ECC. The SISO equalizer is implemented based on a BCJR algorithm [5], whereas the SISO decoder is implemented based on the message passing algorithm [26] with 5 internal iterations. The PLL gain parameters for different iterative timing recovery schemes were optimized based on minimizing σ_ϵ/T at $E_b/N_0 = 5$ dB. Each point in the EXIT chart is obtained by averaging the extrinsic output pdf's over $N_b = 1000$ blocks according to

$$\overline{p(l|C = \pm 1)} = \frac{1}{N_b} \sum_{i=1}^{N_b} p_i(l|C = \pm 1). \tag{49}$$

Note that we observed that the EXIT chart analysis cannot be used to predict the performance of iterative timing recovery schemes when a cycle slip occurs because I_E^{out} will decrease drastically. To demonstrate that all benefits obtained from the EXIT chart analysis are still valid for iterative timing recovery schemes, *only* the data blocks that contain no cycle slip will be used to generate the EXIT chart.

We first consider the system at $E_b/N_0 = 5$ dB with a moderate random walk parameter $\sigma_w/T = 0.5\%$, which implies a low probability of occurrence of a cycle slip. The ξ's for the conventional receiver, the NBM algorithm, and per-survivor iterative timing recovery are 0.0053, 0.0053, and 0.0028, respectively. Figure 40 depicts the EXIT chart of different iterative timing recovery schemes and its corresponding BER plot. The dashed arrows in Figure 40(a) describe the system trajectory of the system with perfect timing (using $\hat{\tau}_k = \tau_k$ to sample $y(t)$). The iteration starts at the $(0, 0)$ point where the equalizer has no *a priori* information so that $I_E^{in} = 0$, and ends at

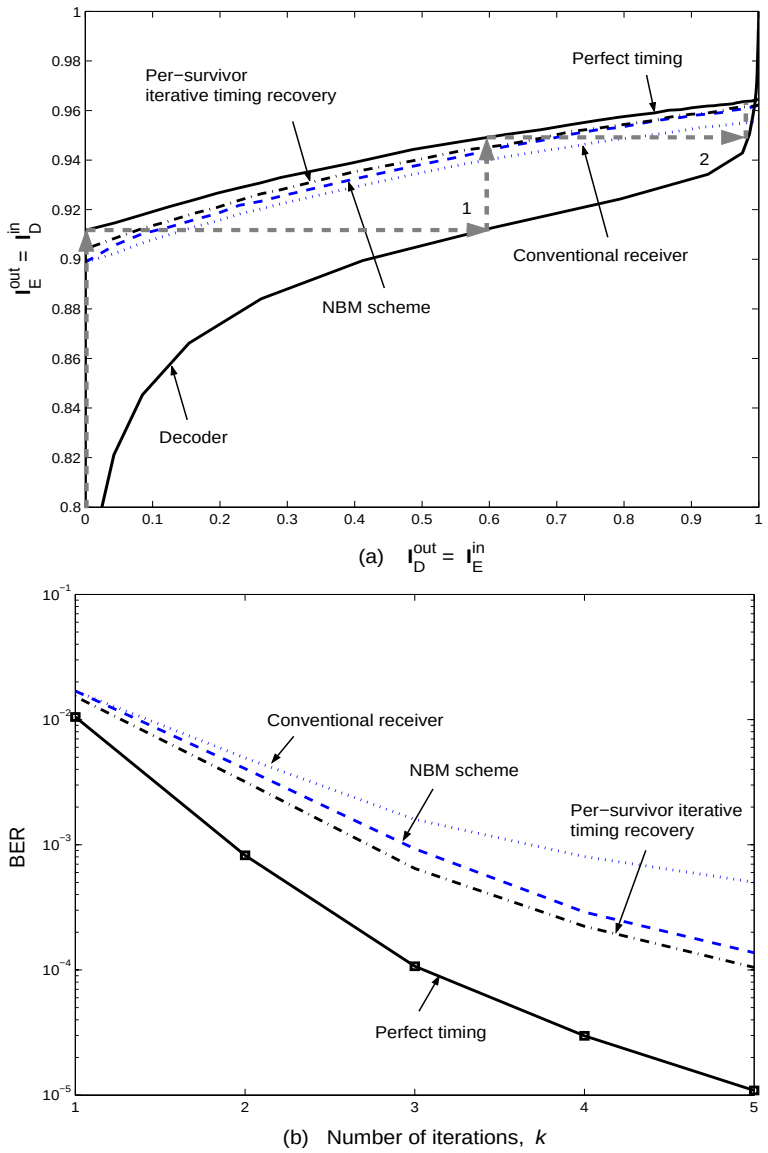


Figure 40: (a) The mutual information transfer characteristics of different iterative timing recovery schemes and (b) their corresponding BER curves at $E_b/N_0 = 5$ dB and $\sigma_w/T = 0.5\%$.

the point where the equalizer transfer characteristic intersects the decoder one. Note that we refer to the first pass through the decoder as the first iteration. As expected, for a given number of iterations, the system with perfect timing yields the largest I_D^{out} or, equivalently, the lowest BER, followed by per-survivor iterative timing recovery, the NBM scheme, and the conventional receiver. Apparently, the higher the I_D^{out}, the

smaller the BER.

We also show the system trajectory of different schemes with two different coded block lengths in Figure 41. Clearly, the longer the coded block length, the better the system trajectory matches the transfer characteristics. This is because the EXIT chart analysis assumes that all LLRs are independent, and thus it is valid only for an infinite block length [80].

Next, we consider the system at $E_b/N_0 = 5$ dB with a severe random walk parameter $\sigma_w/T = 1\%$, which implies a high probability of occurrence of a cycle slip. The ξ's for the conventional receiver, the NBM scheme, and per-survivor iterative timing recovery are 0.0103, 0.0103, and 0.006, respectively. The EXIT chart of different iterative timing recovery schemes and its corresponding BER plot are depicted in Figure 42. Clearly, there is a big performance gap between per-survivor iterative timing recovery and the NBM scheme. This implies that per-survivor iterative timing recovery outperforms the NBM scheme when the timing error is large (same conclusion as presented in Section 4.6).

Finally, Figure 43 plots the E_b/N_0 (in dB) required to achieve BER $= 10^{-4}$ at the decoder output at the 2-nd iteration as a function of σ_w/T's. The solid lines are the predicted BER computed from (48), whereas the dashed lines are the BER obtained by simulating data transmission over the complete iterative receiver. As expected, per-survivor iterative timing recovery performs better than the NBM scheme, and both outperforms the conventional receiver, especially when the timing error is large. Obviously, there is a big gap between the predicted BER and the simulated one. However, we observed that this gap gets smaller when using a larger coded block length. Again, this is because the EXIT chart analysis is based on the assumption that the coded block length is infinite [80]. Nonetheless, for a small coded block length, we can use the predicted BER as a practical bound on the achievable performance.

In addition, we also observed that all benefits obtained from the EXIT chart

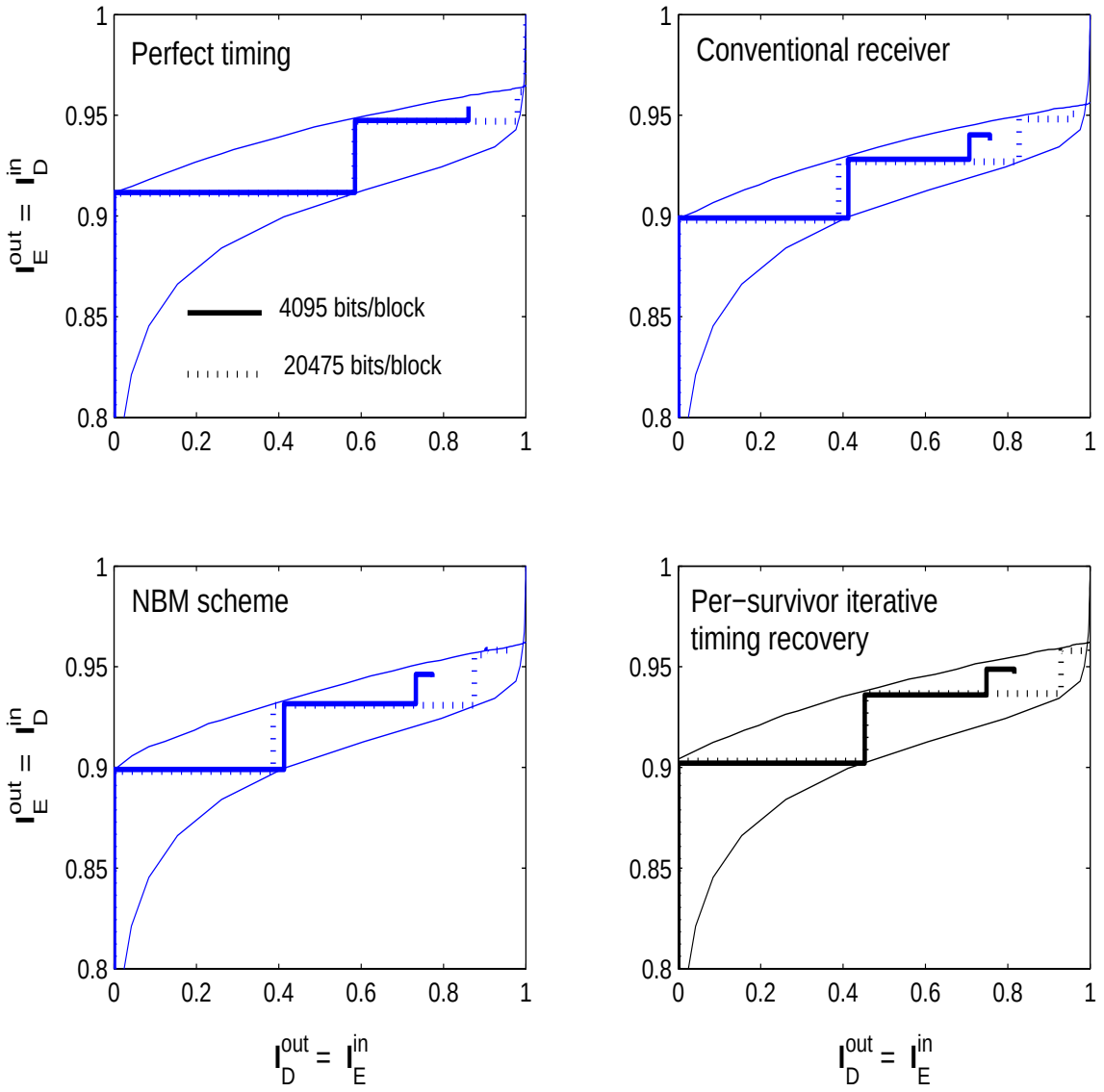


Figure 41: System trajectories of different iterative timing recovery schemes at $E_b/N_0 = 5$ dB and $\sigma_w/T = 0.5\%$, where the solid lines are based on the coded block length of 4095 bits, and the dashed lines are based on the coded block length of 20475 bits.

analysis as investigated in [80] (e.g., finding the SNR threshold, evaluating the effect of the different constituent codes, etc.) are also valid for iterative timing recovery schemes, assuming that there is no cycle slip. Therefore, it is sufficient to use the EXIT chart as a convenient means to compare the performance of different schemes because it requires much less simulation time than a BER criterion.

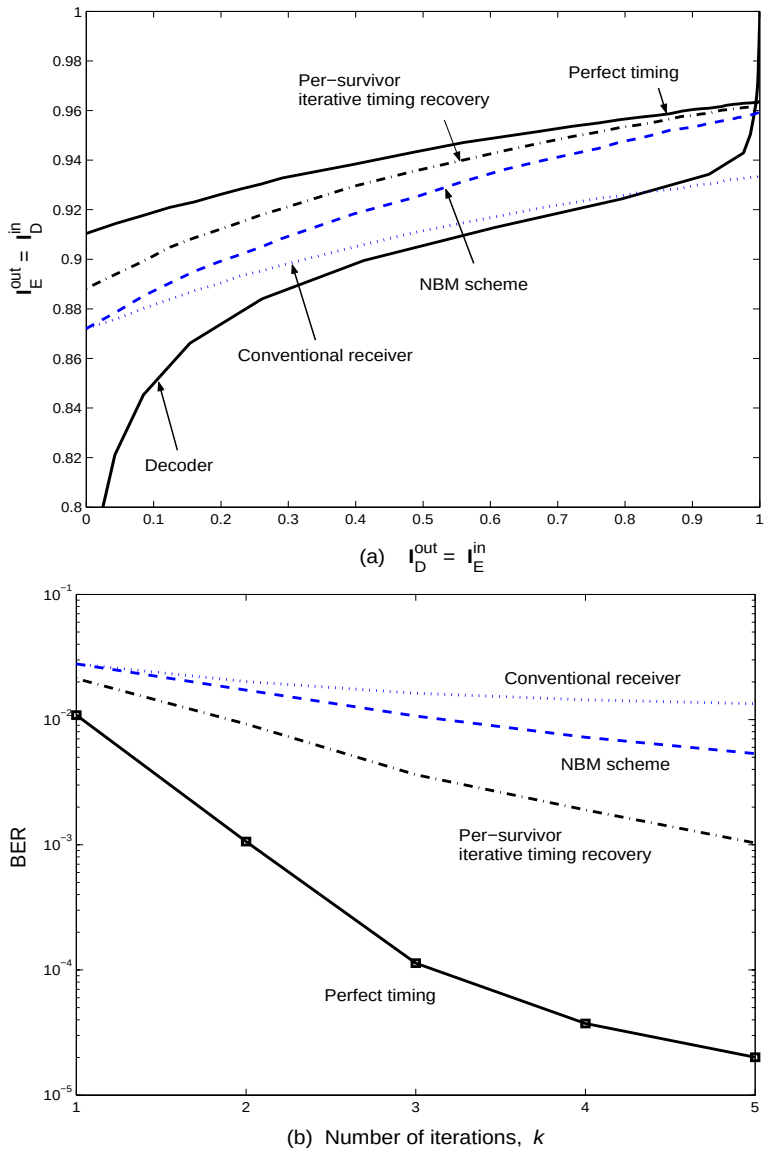


Figure 42: (a) The mutual information transfer characteristics of different iterative timing recovery schemes and (b) their corresponding BER curves at $E_b/N_0 = 5$ dB and $\sigma_w/T = 1\%$.

4.9 Exploring Per-Survivor Iterative Timing Recovery

In this section, we answer some interesting questions related to per-survivor iterative timing recovery, which can be classified as follows.

(i) Why does per-survivor iterative timing recovery perform better than the NBM

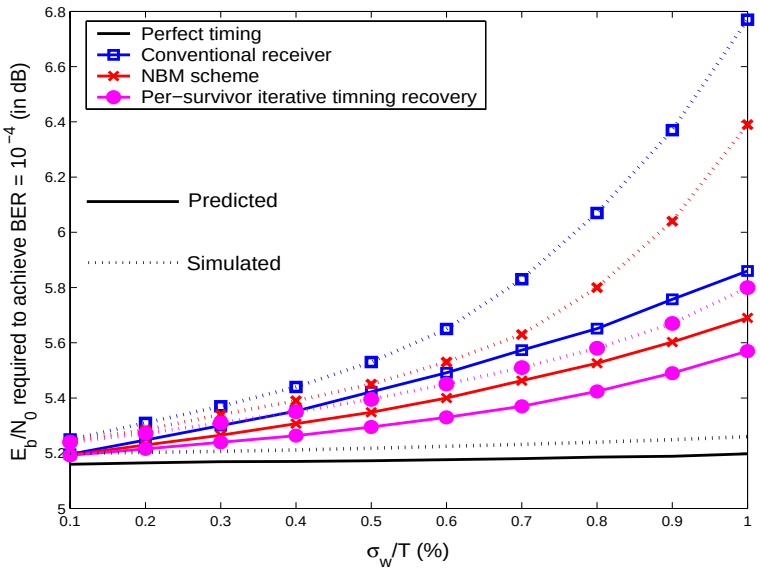


Figure 43: E_b/N_0 (in dB) required to achieve BER $= 10^{-4}$ at the decoder output at the 2-nd iteration as a function of σ_w/T's.

scheme? The reasons for this question might be as follows.

- The front-end PLL used in the NBM scheme does not work well if compared to PSP-based timing recovery. To verify this statement, we run a simulation based on an uncoded PR-IV channel model shown in Figure 4. Figure 44 plots the percentage of occurrence of a cycle slip at $E_b/N_0 =$ 5 dB, where ξ's were optimized based on minimizing σ_ϵ/T for each σ_w/T at $E_b/N_0 = 5$ dB. Clearly, PSP-based timing recovery experiences a fewer number of cycle slips than conventional timing recovery, especially when σ_w/T is large.

- Whenever there is some confidence on the data bits, per-survivor iterative timing recovery tends to correct a cycle slip much more efficiently than the NBM scheme. This can be validated by plotting the probability of an uncorrected cycle slip as a function of the mutual information of the *a priori* information of the input of the SISO equalizer, $\{I(L_D(c_k); c_k)\}$, based on

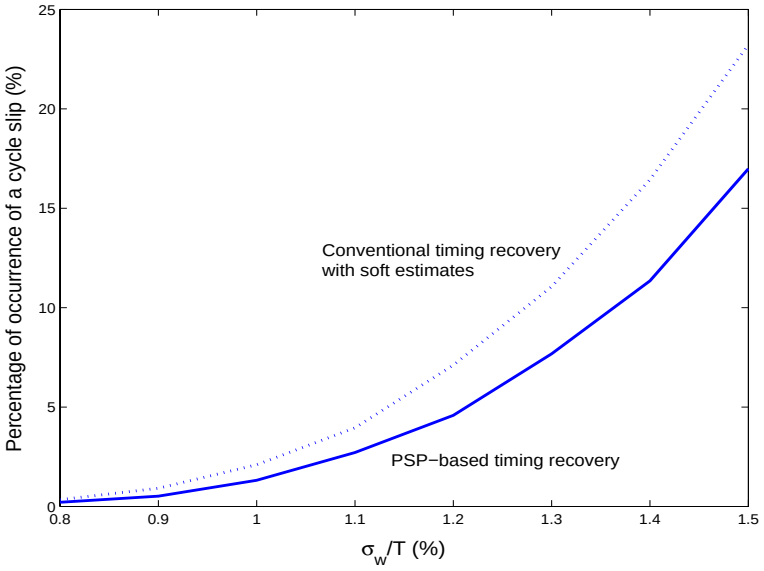


Figure 44: Percentage of occurrence of a cycle slip for an uncoded PR-IV channel at $E_b/N_0 = 5$ dB.

a coded PR-IV channel model (without a precoder) with $\sigma_w/T = 1\%$ and $E_b/N_0 = 5$ dB. We use the same receiver architecture for the NBM scheme as shown in Figure 39. We count the number of cycle slips directly at the output of the SISO equalizer at the first pass (i.e., no need to perform error-correction decoding). It is evident that the NBM scheme can correct a cycle slip only when $I(L_D(c_k); c_k)$ is high enough. This corresponds to the situation that the NBM scheme performs after a large number of iterations. That is why the NBM scheme requires a large number of iterations to provide a good performance when the timing jitter is large.

– The NBM scheme tends to slow down the convergence rate of the system, especially when the confidence of the *a priori* information of the input of the SISO equalizer, $\{L_D(c_k)\}$, is small. To justify this statement, we consider the receiver architecture shown in Figure 39 and assume that all the signs of $\{L_D(c_k)\}$ are correct but their corresponding magnitudes are the parameters that we are considering. The NBM scheme uses $\{L_D(c_k)\}$

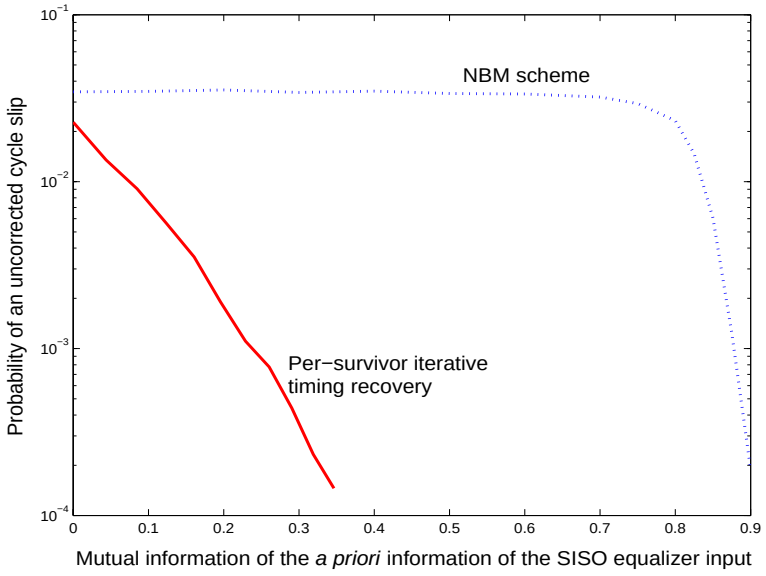


Figure 45: Probability of an uncorrected cycle slip as a function of the mutual information of the *a priori* information of the SISO equalizer input ($\sigma_w/T = 1\%$ and $E_b/N_0 = 5$ dB).

Table 3: Soft estimate computation for a PR-IV channel based on (41).

$\lvert L_D(c_k)\rvert$	$\tilde{r}_k$
0.1	$\{0, \pm 0.0999\}$
1	$\{0, \pm 0.9242\}$
2	$\{0, \pm 1.5232\}$
5	$\{0, \pm 1.9732\}$
10	$\{0, \pm 1.9988\}$

to compute the soft estimates, which will then be used to refine the samples. As shown in Table 3, the NBM scheme needs large $|L_D(c_k)|$ to produce a soft estimate close to the actual value of $\hat{r}_k \in \{0, \pm 2\}$. In other words, we can write $\tilde{r}_k = K\hat{r}_k$, where $0 < K \leq 1$ is a scaling factor determining the confidence of $\{L_D(c_k)\}$. By substituting $\tilde{r}_k = K\hat{r}_k$ in the PLL update equation of the NBM scheme given in (35), one obtains

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \xi_{\text{eff}} \frac{3T}{16} \{y_k \hat{r}_{k-1} - y_{k-1} \hat{r}_k\}, \tag{50}$$

where $\xi_{\text{eff}} = K\xi$, which is always less than or equal to ξ. This suggests

that the NBM scheme will slow down the convergence rate of the system when $|L_D(c_k)|$ is small. Unlike the NBM scheme, we observed that the timing update operation in per-survivor iterative timing recovery is likely to perform on the correct decision even if $|L_D(c_k)|$ is not large enough, as required in the NBM scheme. Therefore, per-survivor iterative timing recovery is less likely to slow down the convergence rate of the system than the NBM scheme.

- We observed that the NBM scheme gradually corrects a cycle slip at the beginning of a cycle slip (from left to right), whereas per-survivor iterative timing recovery randomly corrects a cycle slip as illustrated in Figure 46, where the dots represent the errors occurred at each location in the data packet at the output of the SISO equalizer. The number labeled on each dotted line is the total number of errors occurred at the output of the SISO equalizer. Because the NBM scheme can only correct a cycle slip at the beginning of a cycle slip, it then requires a large number of iterations to correct a cycle slip, especially when a cycle slip occurs at the very beginning of the data packet.

(ii) Why do we average the backward sampling phase offset? Why do we use (B-25) (see Figure 27) as a criterion? The answers to these questions might be as follows.

- We proposed to average the backward sampling phase offset, $\hat{\tau}_k^b(p)$, with the forward one, $\hat{\tau}_k(p)$, because we want to avoid a cycle slip that might happen when $\hat{\tau}_k^b(p)$ starts deviating from $\hat{\tau}_k(p)$. An explanation for this is possible. Suppose we let A be the event that a cycle slip is occurred during forward recursion and B be the event that a cycle slip is occurred during backward recursion. Therefore, if A and B are independent (i.e.,

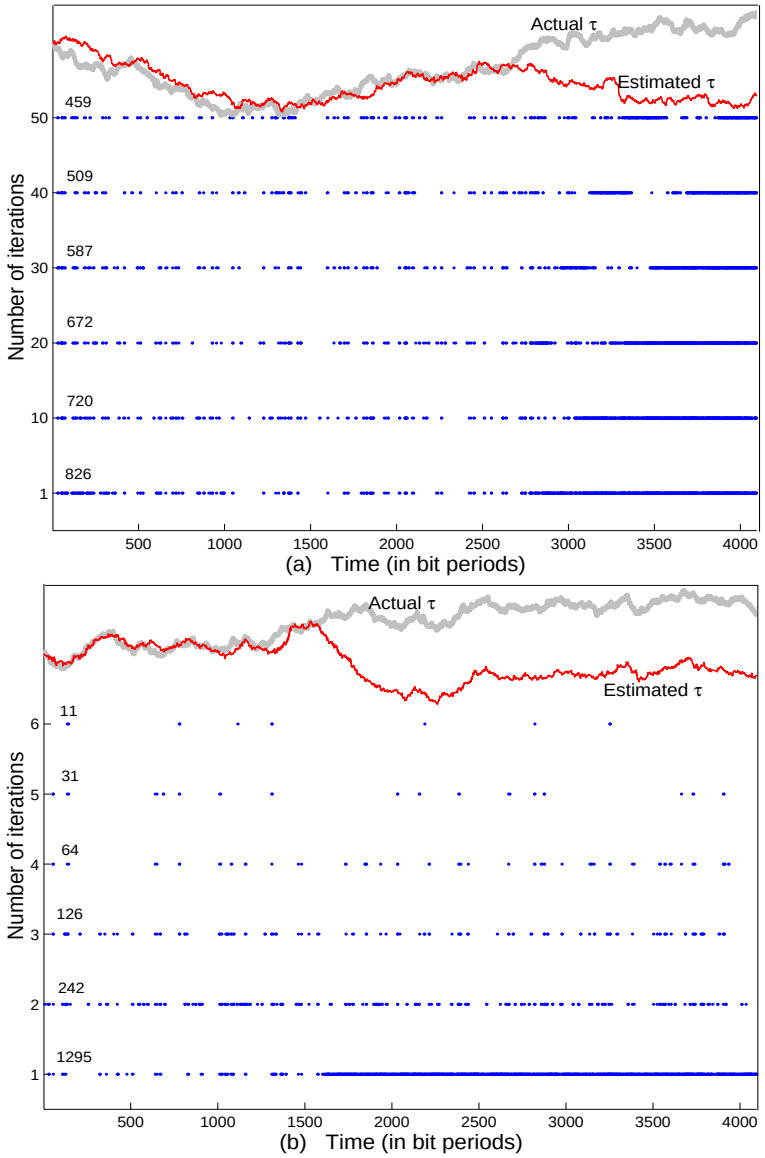


Figure 46: Error positions at each iteration for (a) the NBM scheme and (b) per-survivor iterative timing recovery at $E_b/N_0 = 5$ dB and $\sigma_w/T = 1\%$.

corresponding to without averaging), the probability that these two events are occurred on the same packet can be written as

$$\Pr[A \cup B] = \Pr[A] + \Pr[B]. \tag{51}$$

On the other hand, if A and B are dependent (i.e., corresponding to our averaging criterion), the probability that these two events are occurred will

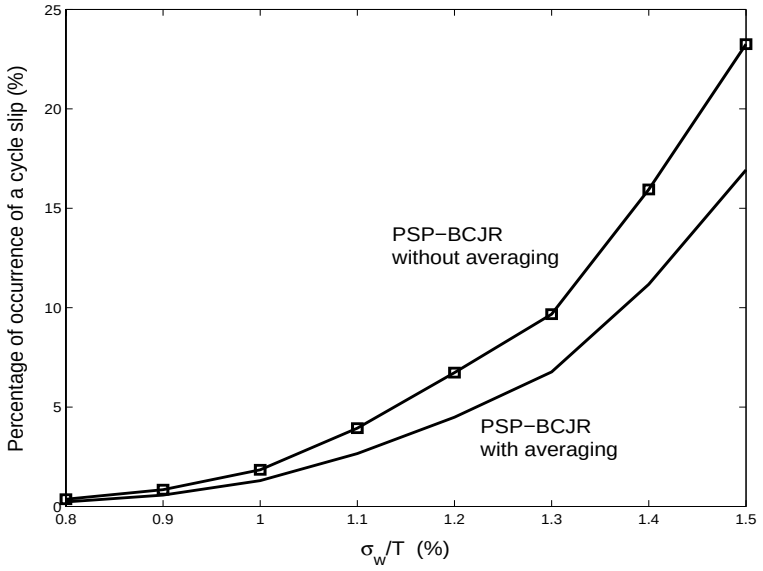


Figure 47: Percentage of occurrence of a cycle slip as a function of σ_w/T's for the precoded PR-IV channel at $E_b/N_0 = 5$ dB.

be

$$\Pr[A \cup B] = \Pr[A] + \Pr[B] - \Pr[A \cap B]. \tag{52}$$

Apparently, (52) is always less than or equal to (51) because $\Pr[A \cap B] \geq 0$. This suggests that averaging the backward sampling phase offset might help reduce the probability of occurrence of a cycle slip, as illustrated in Figure 47.

– Unlike the forward timing update operation where perfect acquisition is assumed, we have no clue about the initial backward sampling phase offset, $\hat{\tau}^b_{L+\nu}(p)$, that is used to start the backward timing update operation. Thus, we initialize it equal to $\hat{\tau}_{L+\nu}(p)$ according to (B-14). Similarly, at each state during the backward timing update operation, we have no knowledge about how reliable the state information ($\alpha_k(p)$ and $\beta_k(p)$) is. Therefore, it is reasonable to assume that they are equally reliable. That is why we average $\hat{\tau}^b_k(p)$ according to (B-25). In addition, we observed that initializing $\hat{\tau}^b_{L+\nu}(p)$ to the correct sampling phase offset does not help improve the

performance of PSP-BCJR significantly.

(iii) Is it a good idea to use the previous set of the timing estimates for the timing update operation at the next iteration? The answer is "no," which might be possibly explained as follows.

- One key factor for the performance of PSP-BCJR is the path metric calculation. This is because if the correct path (guided by the path metric) is chosen, the timing update operation will be fully trained. Since the *a priori* information, λ_k, at the input of the SISO equalizer (see Figure 25) has great influence on the path metric calculation, if λ_k is correct (meaning that it has the same *sign* as the k-th data bit, a_k (see Figure 25), then it is more likely that the correct path will be chosen for the timing update operation. Therefore, we believe that it might be a good idea to rely only on the *a priori* information when performing the timing update operation at each iteration.

- Suppose the previous set of the timing estimates contains a cycle slip. Hence, it will be definitely harmful to use this set of the timing estimates in the timing update operation of the current iteration. Therefore, it is better to rely only on the *a priori* information when performing the timing update operation at each iteration.

4.10 Summary

We proposed a per-survivor version of the BCJR algorithm that performs timing recovery and equalization jointly. With a per-survivor BCJR equalizer, we proposed a per-survivor iterative timing recovery scheme to jointly perform timing recovery, equalization, and error-correction decoding, for coded partial response channels.

Simulation results have shown that per-survivor iterative timing recovery performs

close to the genie-aided receiver, provided that the number of turbo iterations is large enough. Furthermore, per-survivor iterative timing recovery also outperforms the NBM scheme, especially when the timing error is large. This is because per-survivor iterative timing recovery can automatically correct a cycle slip much more efficiently than the NBM scheme. Several possible explanations were given to justify why per-survivor iterative timing recovery performs better than other schemes. In addition, it has been shown that for low to moderate complexity, per-survivor iterative timing recovery still performs better than conventional schemes, especially when operating in a system that experiences a high probability of occurrence of a cycle slip.

We also investigated the performance of a reduced-complexity version of PSP-BCJR. We found that the M- and T- algorithms can be used to reduce the complexity of PSP-BCJR with acceptable performance if their parameters are chosen suitably. Apparently, there is a trade-off between the complexity and the BER performance.

Finally, we have showed that the EXIT chart can be equivalently used instead of BER as a measure to compare the performance of different iterative timing recovery schemes, assuming that there is no cycle slip. Specifically, the system performance predicted by the EXIT chart coincides with that obtained by simulating data transmission over the complete iterative receiver.

CHAPTER 5

APPLICATIONS TO MAGNETIC RECORDING SYSTEMS

This chapter is dedicated to the application of magnetic recording systems. This application is considered because magnetic recording is a primary method of storage for a variety of applications, including desktop, mobile, and server systems. Timing recovery in magnetic recording systems is an increasingly critical problem because of the growing data rate to be supported. Improving the performance of timing recovery will give rise to improved reliability of an entire recording system, which in turn results in increased storage capacity.

The proposed timing recovery schemes presented in Chapters 3 and 4 will be investigated in magnetic recording systems (both longitudinal and perpendicular recording channels [85]) and compared with the conventional schemes used in today's magnetic recording read-channel chip architectures. This experiment will help us decide whether or not the proposed schemes can be feasibly employed in real-life applications as compared to conventional ones.

5.1 Background on Digital Magnetic Recording Systems

Digital magnetic recording has been employed in many applications, including hard disk drives, floppy disk drives, and tape drives. However, all applications are based on the same fundamental principle, which involves a magnetic head and a recording medium, as shown in Figure 48. An inductive head consists of a horseshoe-shaped soft magnetic material with low coercivity and high permeability [85] around which coils

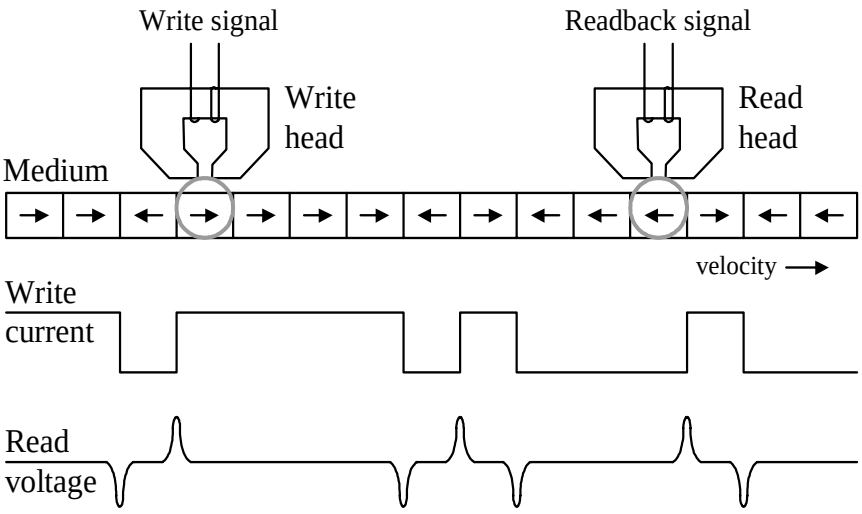


Figure 48: Schematic principle of magnetic recording.

are wound, and a recording medium that is normally comprised of a hard magnetic material with high coercivity.

Two modes of magnetic recording are considered in this work, namely, *longitudinal recording* [9] and *perpendicular recording* [77]. Today's hard disk drive technology is based on longitudinal recording in which the medium magnetization is parallel to the disk plane, as shown in Figure 48. Recently, research on perpendicular recording in which the medium magnetization is perpendicular to the disk plane has been of increasing interest because of the potential for increased storage capacity [77]. It is expected that perpendicular recording will be employed in the next generation of hard disk drives.

5.1.1 Write Process

During the write process, the data bits are converted into a rectangular current waveform called a write current (see Figure 48). This write current is applied to the windings of the write head to produce a magnetic write field in the medium near the head gap. The write field must be larger than the medium coercivity to magnetize the medium along the field direction. By switching the direction of the write field (or

the write current), magnetization transitions can be written in the medium.

Commercial digital recording systems normally employ binary saturation recording, i.e., the magnetization saturated on the medium in only one direction or the opposite. This is because if more than two data levels were recorded, nonlinearities would cause a major problem and signal-to-disturbance ratios would diminish considerably [9].

5.1.2 Read Process

During the read process, the read head senses the change in the flux via the transitions of the magnetization pattern, resulting in an induced voltage pulse in the coil because of Faraday's law. For an isolated transition, the read head produces a read voltage pulse, $g(t)$, or its inverse, $-g(t)$, depending on the direction of the transition (see Figure 48). The pulse $g(t)$ is commonly known as the *transition response* [9], which has a finite amplitude and a finite half-amplitude pulse width.

The transition response for longitudinal recording (also known as the Lorentzian pulse) is given by [9]

$$g(t) = \frac{1}{1 + \left(\frac{2t}{\mathrm{PW}_{50}}\right)^2}, \tag{53}$$

where PW_{50} determines the width of $g(t)$ at half of its peak value. For perpendicular recording, we are interested in a transition response of the form [69]

$$g(t) = \mathrm{erf}\left(\frac{2t\sqrt{\ln 2}}{\mathrm{PW}_{50}}\right), \tag{54}$$

where $\mathrm{erf}(\cdot)$ is an error function defined by $\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$ [42], and PW_{50} determines the width of the derivative of $g(t)$ at half its maximum.

In the context of magnetic recording, the ratio $\mathrm{ND} = \mathrm{PW}_{50}/T$ (where T is the bit duration) represents a *normalized recording density* [9], which defines how many data bits can be packed within the resolution unit PW_{50}. The transition responses of longitudinal and perpendicular recording channels are plotted in Figure 49 for

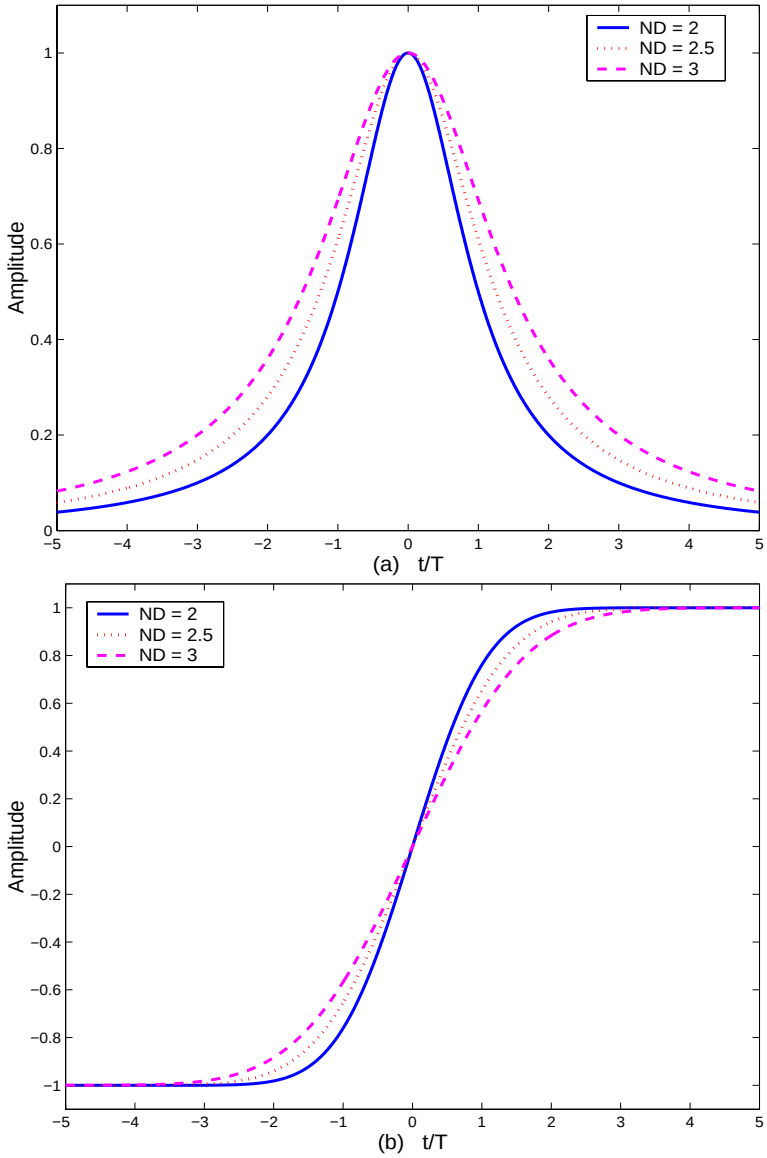


Figure 49: Transition responses for (a) longitudinal and (b) perpendicular recording.

different NDs. Clearly, the transition response spans many symbol intervals as ND increases. This implies that the effect of ISI becomes more severe as ND increases.

Additionally, the response of the head to an isolated bit is commonly known as the *dibit response* [9], which is expressed as $m(t) = g(t) - g(t-T)$. It is easy to show that the frequency response of $m(t)$ for longitudinal recording is given by

$$M(\Omega) = \exp(-\pi|\Omega|\mathrm{ND}) \cdot \{1 - \exp(-j2\pi\Omega)\}, \tag{55}$$

whereas for perpendicular recording it is

$$M(\Omega) = \frac{T}{j\pi\Omega} \cdot \exp\left(-\frac{\pi^2\Omega^2 \mathrm{ND}^2}{\ln 16}\right) \cdot \{1 - \exp(-j2\pi\Omega)\}, \tag{56}$$

where $\Omega = fT$ is a normalized frequency variable, f is a frequency variable in Hertz, $|x|$ takes on the absolute value of x, and $j = \sqrt{-1}$ is an imaginary number. Figure 50 shows the normalized frequency responses of the dibit responses for different ND's. Apparently, the signal energy becomes more concentrated at low frequencies as ND increases for both channels. Furthermore, a longitudinal recording channel exhibits a spectral null at d.c., while a perpendicular recording channel contains a d.c. component.

5.1.3 Magnetic Recording Channel Model

A magnetic recording system can also be expressed in terms of a mathematical model, as depicted in Figure 51. This system model will be referred to as a *realistic* channel model because it represents all the components that are employed in magnetic recording read-channel chip architectures.

A binary input data sequence $a_k \in \{\pm 1\}$ with bit period T is filtered by an ideal differentiator $1 - D$, where D is the delay operator, to form a transition sequence, $b_k \in \{-2, 0, 2\}$, where $b_k = \pm 2$ corresponds to a positive or a negative transition, and $b_k = 0$ corresponds to the absence of a transition. The transition sequence b_k passes through the channel represented by the transition response $g(t)$ and is corrupted by the noise $v(t)$. The readback signal, $p(t)$, is filtered by a low-pass filter (LPF) to eliminate the out-of-band noise and is sampled at the instants controlled by the timing recovery block. Then, the sampler output is equalized by an equalizer so that the equalizer output resembles the desired sample. Eventually, the symbol detector performs ML equalization to determine the most likely input sequence.

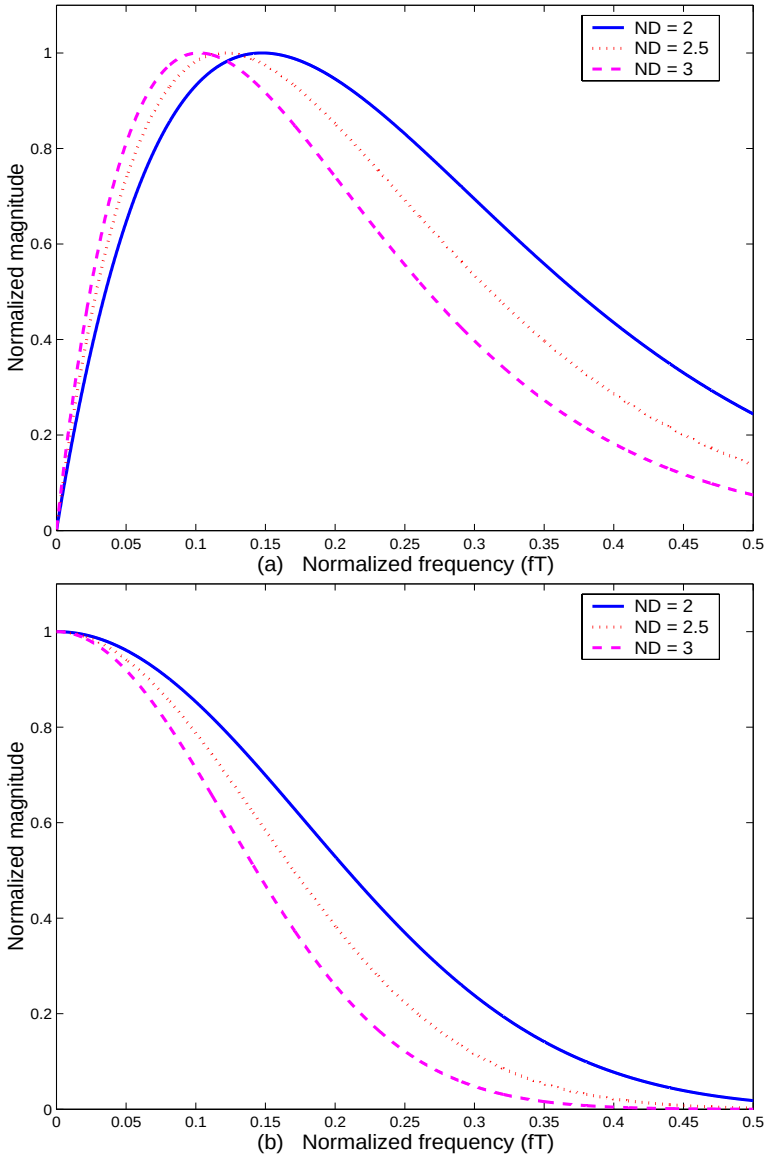


Figure 50: Frequency responses of the dibit responses for (a) longitudinal and (b) perpendicular recording.

5.2 Equalization and Target Design

A widely used symbol detector in magnetic recording systems is a Viterbi detector [24]. Because the complexity of the Viterbi detector grows exponentially with channel memory, the equalizer is usually employed to shape the overall channel impulse response into a shorter response called the *target response* [9, 53], $H(D)$, thus reducing the complexity of the Viterbi detector. The technique of using the equalizer

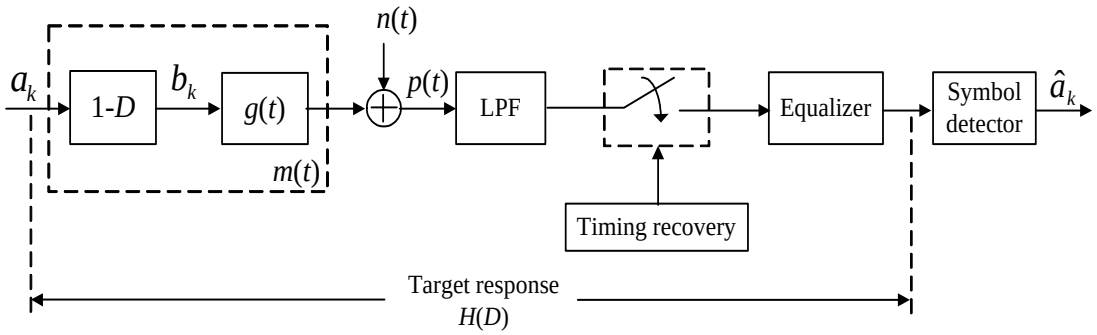


Figure 51: A realistic magnetic recording channel model.

in conjunction with the Viterbi detector is commonly known as a *partial response maximum-likelihood* (PRML) technique [9, 15], which is practically utilized in magnetic recording systems. This is done in two steps. First, the received signal is equalized to a PR target whose response is as close to a channel response as possible. Then, the Viterbi detector performs ML equalization on the resulting PR trellis.

The generally accepted PR target [81] for longitudinal recording is of the form $H(D) = (1 - D)(1 + D)^n$ [9], whereas the PR target for perpendicular recording is $H(D) = (1 + D)^n$ [39], where n is an integer. Apparently, the term $(1 - D)$ is not needed for perpendicular recording because the perpendicular recording channel contains a d.c. component. Figure 52 compares the frequency responses of different targets. It is clear that as ND increases, a larger value of n is required because the effect of ISI becomes more severe at high ND. Hence, at high ND, a longer target allowing more controlled ISI will provide a better match to the channel response than a shorter target.

By introducing the target response with non-integer valued coefficients, commonly known as a *generalized partial response* (GPR) target [53], the performance gain can be substantially improved, especially at high ND. The choice of the target is crucial because it governs the noise variance at the input to the Viterbi detector. There exist many criteria proposed in the literature for designing a suitable target, such as

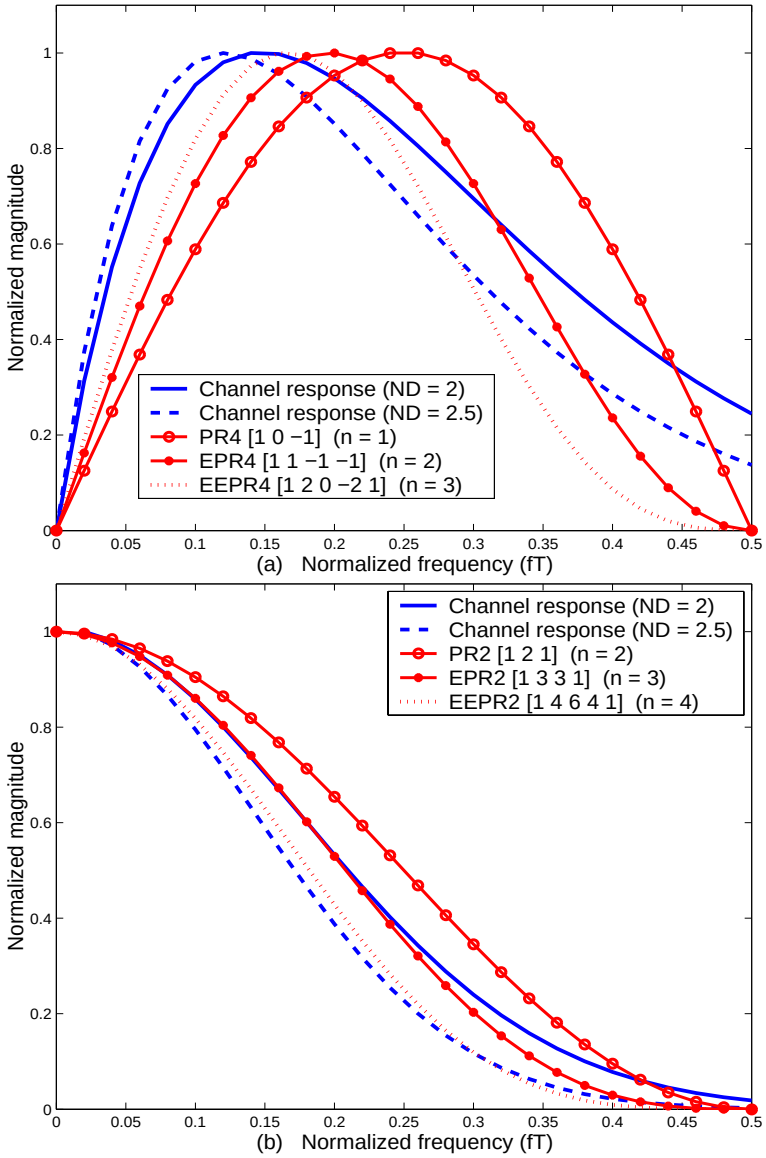


Figure 52: Frequency responses of different targets for (a) longitudinal and (b) perpendicular recording channels.

(i) minimizing the noise power at the output of the equalizer [47]; (ii) maximizing the effective signal-to-noise ratio ($\mathrm{SNR}_{\mathrm{eff}}$) [23, 53]; (iii) minimizing the mean-squared error (MSE) between the equalizer output and the desired target output [53]; and (iv) matching the time or frequency domain of the transition or dibit response of the channel [59]. Nevertheless, only the minimum mean-squared error (MMSE) approach [53] is considered in this work because it is more practical to apply in real-life

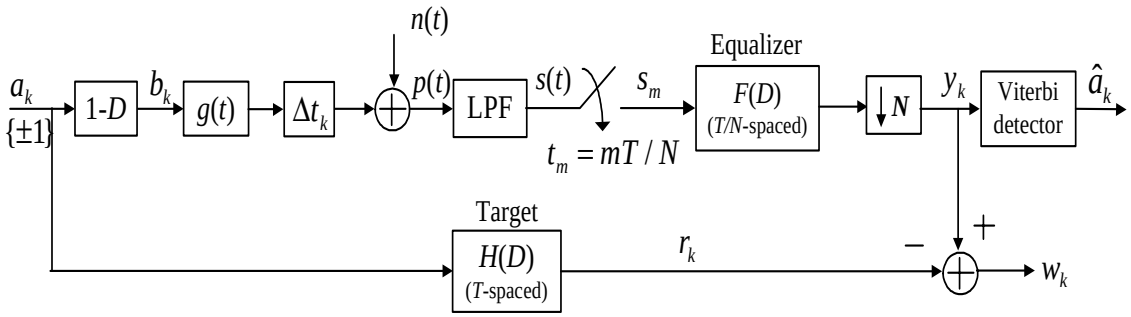


Figure 53: MMSE target design.

applications.

5.2.1 MMSE Target Design

The MMSE approach yields significantly different results depending on which the specific constraint is chosen. We concentrate only on the *monic* constraint [8] because it yields the best performance among other constraints [53]. Consider the system model for the target design in Figure 53. The aim of a finite impulse response (FIR) equalizer, $F(D)$, is to generate the output samples $\{y_k\}$ closely resembling the desired samples $\{r_k\}$ without excessive noise enhancement. This can be achieved by minimizing the MSE between $\{y_k\}$ and $\{r_k\}$.

Let $\mathbf{H} = [h_0 \; h_1 \cdots h_\nu]^{\mathrm{T}}$ represent a $(\nu+1)$-tap T-spaced target and let $\mathbf{F} = [f_{-K} \cdots f_0 \cdots f_K]^{\mathrm{T}}$ represent an M-tap T/N-spaced equalizer ($M = 2K+1$), where h_k and f_k denote the filter coefficients of $H(D)$ and $F(D)$, respectively, $N \in \{1, 2\}$ is an oversampling rate, and $[\cdot]^{\mathrm{T}}$ represents the transpose operation. For simplicity, we assume that K is divisible by N. Therefore, the target and its corresponding equalizer are designed by minimizing

$$E[w_k^2] = E[\{(s_{Nk} * f_k) - (a_k * h_k)\}^2] \tag{57}$$

subject to the monic constraint (i.e., $h_0 = 1$), where $*$ denotes the convolution operator, and $E[\cdot]$ is the expectation operator. In this work, $K = 10$ is employed with the

center tap at $k = 0$. The minimization process yields [53]

$$\lambda = \frac{1}{\mathbf{I}^{\mathrm{T}}(\mathbf{A} - \mathbf{P}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{P})^{-1}\mathbf{I}} \tag{58}$$

$$\mathbf{H} = \lambda(\mathbf{A} - \mathbf{P}^{\mathrm{T}}\mathbf{S}^{-1}\mathbf{P})^{-1}\mathbf{I} \tag{59}$$

$$\mathbf{F} = \mathbf{S}^{-1}\mathbf{P}\mathbf{H}, \tag{60}$$

where λ is the Lagrange multiplier, $\mathbf{I}$ is the $(\nu+1)$-element column vector whose first element is one and the rest is zero, $\mathbf{A}$, $\mathbf{S}$, and $\mathbf{P}$ are $(\nu+1)$-by-$(\nu+1)$, M-by-M, and M-by-$(\nu+1)$ matrices with the (i,j)-th element given by

$$\mathbf{A}(i,j) = E\left[\sum_{k=0}^{L-1} a_{k-i}a_{k-j}\right] \tag{61}$$

$$\mathbf{S}(i,j) = E\left[\sum_{k=0}^{L-1} s_{Nk+K-i}s_{Nk+K-j}\right] \tag{62}$$

$$\mathbf{P}(i,j) = E\left[\sum_{k=0}^{L-1} s_{Nk+K-i}a_{k-j}\right], \tag{63}$$

respectively, and L is the length of the input sequence a_k. The resulting target $\mathbf{H}$ is known as the GPR target. Note that the resulting equalizer with $N = 1$ is called a T-spaced equalizer, whereas that with $N \neq 1$ is known as a *fractionally-spaced equalizer* [63, 82]. As can be seen from the definitions of the matrices, equations (58) – (60) reduce to their counterparts in [53] when $N = 1$.

It should be noted that, in some applications, we are provided with a target but are not given its corresponding equalizer filter. Fortunately, we can still apply the MMSE approach to design the equalizer. The resulting equalizer $F(D)$ can be obtained by substituting the given target response $H(D)$ in (60).

5.2.2 Effective SNR

When comparing the performance of different targets, BER is an ultimate indicator of performance. However, determining BER, especially when BER is less than 10^{-6}, requires a considerable amount of computation time. Instead, the *effective SNR* ($\mathrm{SNR}_{\mathrm{eff}}$) [6] can be considered as an effective approach to determine which target

is the best, because it correlates well with the BER and it can be computed much faster than BER. To compute $\mathrm{SNR}_{\mathrm{eff}}$, we need to determine the dominant *error event* [12, 24, 50] as well as the autocorrelation matrix of w_k, $\mathbf{R}_{\mathrm{ww}}$. This can be accomplished by using only one data sector, as opposed to several data sectors required for the computation of BER. Note that the larger the sector length, the more reliable the result.

The $\mathrm{SNR}_{\mathrm{eff}}$ is defined as [6]

$$\mathrm{SNR}_{\mathrm{eff}} = \frac{(\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon})^2}{\boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{R}_{ww}\boldsymbol{\varepsilon}} = \frac{d_{effmin}^2}{\sigma_w^2}, \tag{64}$$

where $\boldsymbol{\varepsilon}$ is a column vector of the dominant error event. For example, if the dominant error event is such that $\varepsilon(D) = 1 - 2D + 3D^2$, then $\boldsymbol{\varepsilon} = [1, -2, 3]^T$. Let the error sequence $\varepsilon_a(D) = a_1(D) - a_2(D)$, where $a_1(D)$ and $a_2(D)$ are two input sequences of the same length. The error event is then defined as $\varepsilon(D) = \varepsilon_a(D)H(D)$. The performance of the Viterbi detector is largely determined by the error sequence $\varepsilon_a(D)$ that results in the error event $\varepsilon(D)$ having the smallest effective distance, d_{effmin}, rather than the Euclidean distance [6]. The error event $\varepsilon(D)$ and error sequence $\varepsilon_a(D)$ having the smallest effective distance is referred to as the dominant error event and dominant error sequence, respectively.

5.2.3 Numerical Results and Discussion

Consider the system model depicted in Figure 53 with $N = 1$ and perfect synchronization. Because many previous works have been done in the literature for a longitudinal recording channel [9, 53], we then focus on a perpendicular recording channel, where its transition response is given in (54). The *media jitter noise*, Δt_k, is modeled as a random shift in the *transition position* with a Gaussian probability distribution function with zero mean and variance $|b_k/2|\sigma_j^2$ (i.e., $\Delta t_k \sim \mathcal{N}(0, |b_k/2|\sigma_j^2)$) truncated to $T/2$, where $|x|$ takes on the absolute value of x, and σ_j is specified as a percentage of T.

The readback signal, $p(t)$, can be written as

$$p(t) = \sum_{k=-\infty}^{\infty} b_k g(t - kT + \Delta t_k) + n(t), \tag{65}$$

where $n(t)$ is additive white Gaussian noise (AWGN) with two-sided power spectral density $N_0/2$. The readback signal $p(t)$ is filtered by a seventh-order Butterworth low-pass filter[1], whose cutoff frequency is at $1/(2T)$, and then is sampled at time $t_k = kT$. The received sequence, s_k, is equalized so that the output sequence, y_k, resembles the desired sequence, r_k. Eventually, the Viterbi detector performs ML equalization to determine the most likely input sequence.

We define the electronics SNR (or, simply, SNR) as

$$\text{SNR} = 10 \log_{10} \left(\frac{V_p^2}{\sigma_n^2} \right) \quad \text{(dB)} \tag{66}$$

where $V_p = g(\infty) = 1$ is the peak amplitude of an isolated transition pulse and $\sigma_n^2 = N_0/(2T)$ is the input AWGN power. Each BER point was computed using as many 4096-bit data sectors as needed to collect 1000 error bits, while each SNR_{eff} point was computed using only one data sector. For convenience, we denote the "GPRn" target as the n-tap GPR target with the monic constraint. For each ND, the SNR used to design the target and its corresponding equalizer was chosen to minimize the SNR required to achieve the desired BER.

5.2.3.1 BER Performance

Figure 54(a) compares the performance of different targets as a function of NDs in the absence of media jitter noise (i.e., $\sigma_j/T = 0\%$). As illustrated, GPR targets can outperform PR targets, especially at higher NDs. This is because the GPR target provides a better match to the channel response than the PR targets. In Figure 54(b), we pick ND = 2.5, and this time compare the performance of different targets as a

[1]Since most of the signal energy is confined within $|f| \leq 1/(2T)$, a low-pass filter also provides the sufficient statistic [52].

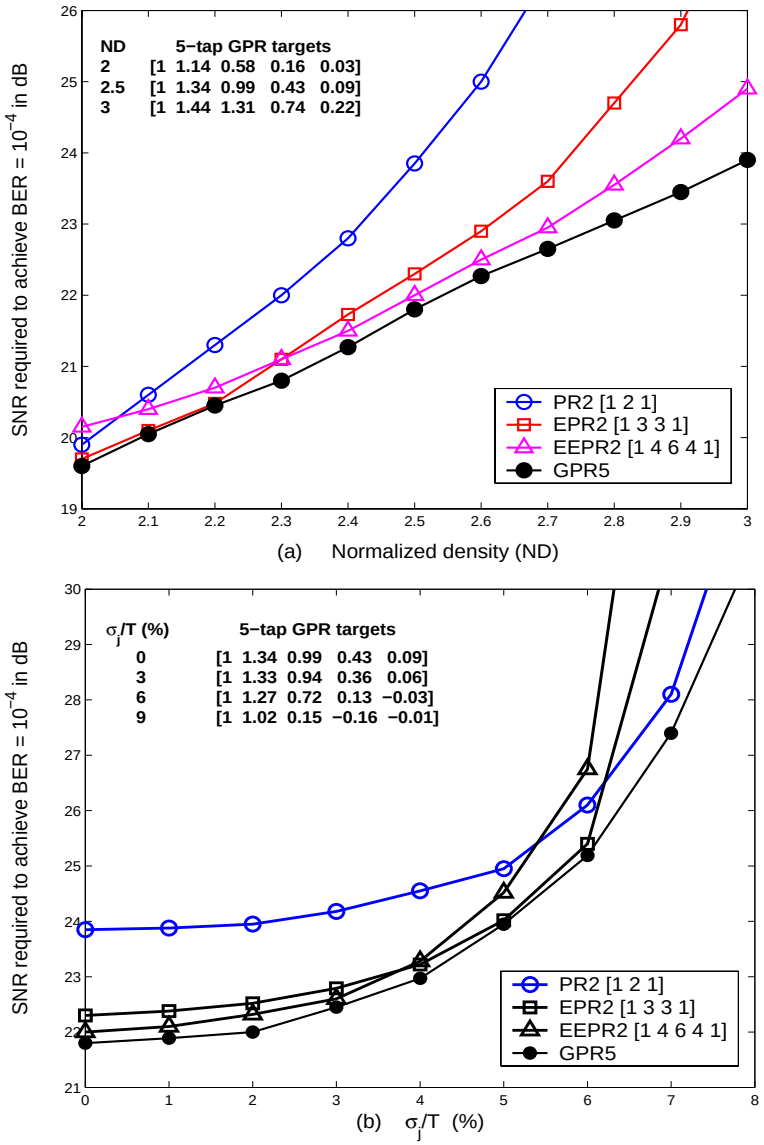


Figure 54: (a) Required electronics SNR vs. ND without media jitter noise, and (b) required electronics SNR vs. σ_j/T at ND = 2.5.

function of σ_j/T's. Again, it is clear that the GPR target requires a lower SNR to achieve BER $= 10^{-4}$ than PR targets for all σ_j/T's. We would like to point out that, even though the PR2 target (i.e., $H(D) = 1 + 2D + D^2$) requires a lower SNR than longer PR targets when σ_j/T is large (which might be because the PR target having a fewer number of coefficients is less sensitive to the media jitter noise than that having a larger number of coefficients), this is not the case for the GPR targets because we

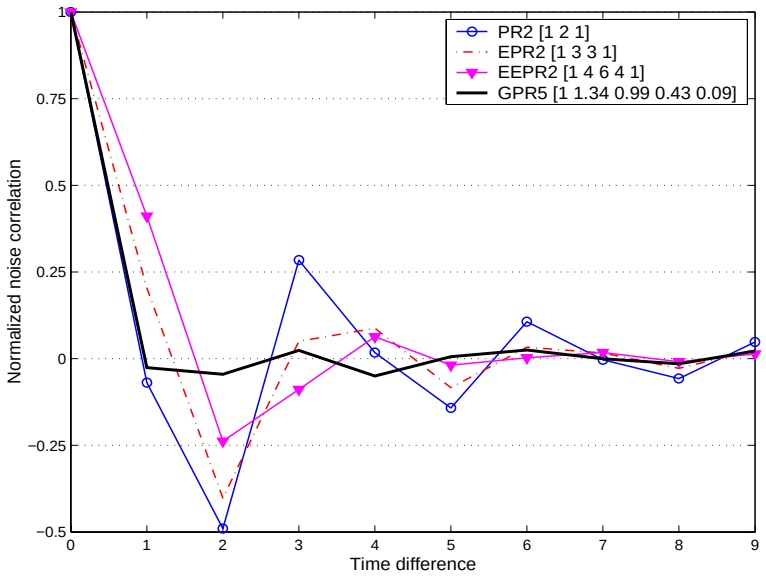


Figure 55: Noise correlation of different targets at the input of the Viterbi detector for ND = 2.5, $\sigma_j/T = 0\%$,and SNR = 22 dB.

observed that they still provide a good performance as the target length increases.

Another reason that the GPR target performs better than other targets is because the GPR target tends to whiten the noise at the input of the Viterbi detector. This can be verified by plotting the noise correlation of different targets at the input of the Viterbi detector in Figure 55. Apparently, the GPR5 target seems to whiten the noise at the input of the Viterbi detector.

5.2.3.2 Error Event Characterization

We also investigate the error events for the perpendicular channel. Table 4 shows the error sequences and their relative frequency of occurrence for the system operating at ND = 2.5 and BER = 10^{-4}. Note that "+" represents "2" and "-" denotes "-2", and all error sequences have a corresponding symmetrical sequence, i.e., $\boldsymbol{\varepsilon}_a = -\boldsymbol{\varepsilon}_a$.

For low σ_j/T cases (0 to 3%), the dominant error sequence for longitudinal recording was shown to be {2, -2, 2} [53], while we found that the dominant error sequence

Table 4: Error sequences for different targets and σ_j/T's at ND = 2.5 and BER = 10^{-4}.

Error sequences	PR2	GPR5	GPR5	GPR5	GPR5
ε_a	σ_j/T=0%	σ_j/T=0%	σ_j/T=3%	σ_j/T=6%	σ_j/T=9%
+	4.90%	3.19%	3.36%	5.84%	41.53%
+-	**67.54**%	**83.25**%	**79.66**%	35.21%	9.62%
+-+	5.79%	0.35%	1.98%	38.31%	21.53%
+-+-	0.51%	0.58%	1.14%	6.66%	15.73%
+-+-+	0.13%	0.23%	0.48%	2.66%	6.31%
+-0+-	15.53%	8.75%	8.94%	3.03%	0.00%
+-0+-0+-	1.34%	0.73%	0.84%	0.22%	0.00%
Others	4.26%	2.72%	3.60%	8.06%	5.28%

for perpendicular recording, for all targets, is {2, -2} [39, 40]. Additionally, the number of dominant error sequences tends to increase as σ_j/T increases. Performance can be further improved by designing and utilizing codes that avoid all dominant error sequences [17]. Another significant point is that because of the different nature of error events, post-processors that work well with longitudinal recording might not work as well with perpendicular recording.

It is well-known that if the noise at the input of the Viterbi detector is white, the performance of the Viterbi detector will then be largely determined by the error sequence that has the minimum squared Euclidean distance, d^2_{min}, given by [24]

$$d^2_{min} = \min_{\boldsymbol{\varepsilon}_a}\{d^2(\boldsymbol{\varepsilon}_a)\}, \tag{67}$$

where

$$d^2(\boldsymbol{\varepsilon}_a) = \boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon}. \tag{68}$$

However, since the noise resulting from designing the GPR target is in general colored noise, it is better to use the squared effective distance, $d^2_{eff}(\boldsymbol{\varepsilon}_a)$, which is given by [12]

$$d^2_{eff}(\boldsymbol{\varepsilon}_a) = \sigma^2_w \frac{(\boldsymbol{\varepsilon}^{\mathrm{T}}\boldsymbol{\varepsilon})^2}{\boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{R}_{ww}\boldsymbol{\varepsilon}}, \tag{69}$$

Table 5: Error event characterization of the PR2 and GPR5 targets at ND = 2.5 and $\sigma_j/T = 0\%$.

Error sequence $\boldsymbol{\varepsilon}_a$	PR2 [1 2 1]			GPR5 [1 1.34 0.99 0.43 0.09]		
	$d^2(\boldsymbol{\varepsilon}_a)$	$d^2_{eff}(\boldsymbol{\varepsilon}_a)$	Percentage of occurrence	$d^2(\boldsymbol{\varepsilon}_a)$	$d^2_{eff}(\boldsymbol{\varepsilon}_a)$	Percentage of occurrence
+	24	32.70	4.90%	15.87	16.72	3.19%
+-	16	**12.09**	67.54%	6.70	**6.76**	83.25%
+-+	16	16.21	5.79%	10.77	11.44	0.35%
+-+-	16	18.63	0.51%	10.44	10.62	0.58%
+-+-+	16	18.00	0.13%	10.82	11.13	0.23%
+-0+-	24	**14.06**	15.53%	8.25	**8.53**	8.75%
+-0+-0+-	32	17.09	1.34%	9.80	9.94	0.73%
Others			4.26%			2.72%

to realistically measure the system performance or to characterize the error event. We verify this statement by characterizing the error event based on the PR2 and GPR5 targets in Table 5 with ND = 2.5 and $\sigma_j/T = 0\%$. It is clear that $d^2_{eff}(\boldsymbol{\varepsilon}_a)$ more accurately predicts which error sequence is most likely to occur in the system than $d^2(\boldsymbol{\varepsilon}_a)$. For example, if we use $d^2(\boldsymbol{\varepsilon}_a)$ as a criterion, we cannot determine which error sequence is most likely to occur because there are several error sequences having the same $d^2_{min}(\boldsymbol{\varepsilon}_a) = 16$. Nevertheless, if we use $d^2_{eff}(\boldsymbol{\varepsilon}_a)$ as a criterion, it is apparent that the error sequence {+ -} is most likely to occur because it has the smallest $d^2_{eff}(\boldsymbol{\varepsilon}_a)$. The same result is also valid for the GPR5 target.

Therefore, the most likely error sequence is the one that yields the smallest effective distance, not the smallest Euclidean distance. Furthermore, the error sequence having the smallest $d^2_{eff}(\boldsymbol{\varepsilon}_a)$ is not necessarily to be the same error sequence that has the smallest $d^2(\boldsymbol{\varepsilon}_a)$. Note that this result is true provided that the system of interest is evaluated at moderately high SNRs, where there is only one dominating error event.

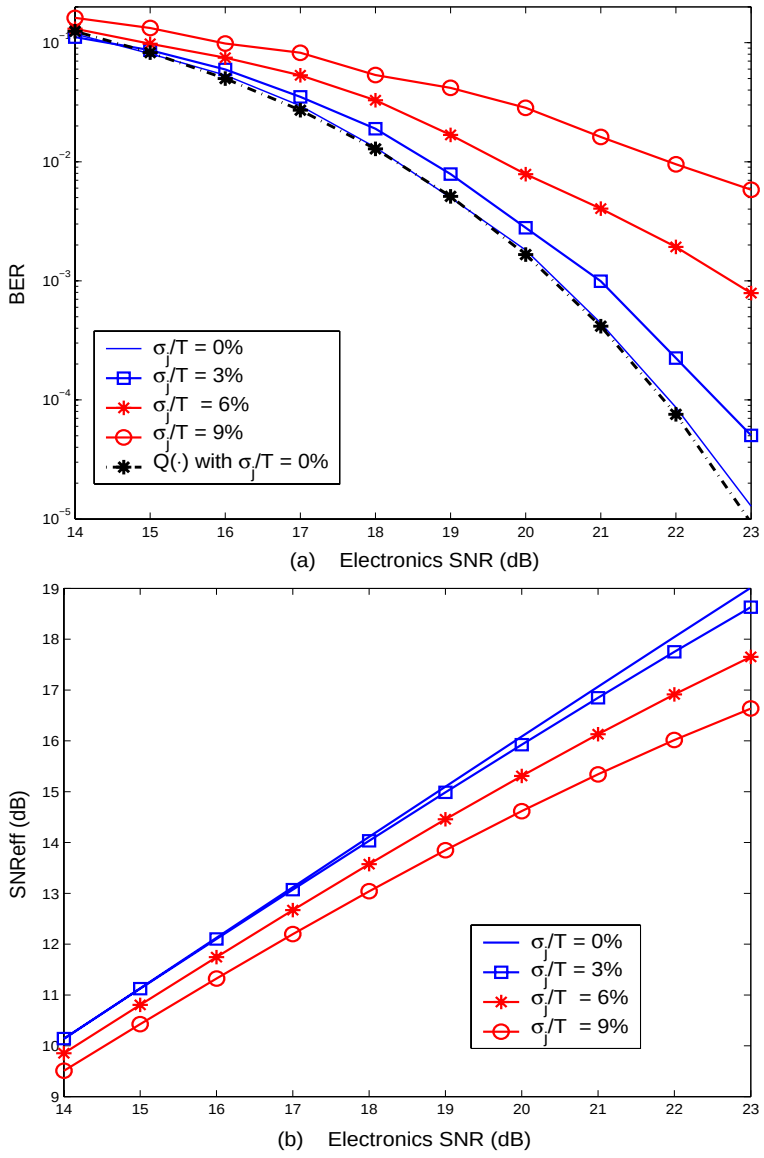


Figure 56: (a) BER and (b) SNR_{eff} performances of the GPR5 target at ND = 2.5.

5.2.3.3 Effective SNR Performance

We now illustrate the fact that BER and SNR_{eff} correlate well, especially when σ_j/T is low (this might not be true as σ_j/T increases because there will be more than one dominant error sequences, and our SNR_{eff} definition does not take this into account). The BER and SNR_{eff} performances of the GPR5 target are compared in Figure 56 at ND = 2.5. Clearly, the SNR_{eff} performance coincides with the BER performance.

Figure 56(a) also shows that, at low σ_j/T, $\mathrm{SNR}_{\mathrm{eff}}$ can be used to estimate the BER according to BER $\approx AQ(\frac{1}{2}\sqrt{\mathrm{SNR}_{\mathrm{eff}}})$ [53], where A is a constant independent of σ_w^2, and $Q(x) = \frac{1}{2\pi}\int_x^\infty e^{-t^2/2}dt$ is the tail integral of the Gaussian density function [42]. For instance, at $\sigma_j/T = 0\%$, the estimated BER labeled as "Q(·)" is in agreement with the actual BER obtained from simulation when $A = 2.3$.

In Figure 57(a), the BER versus $\mathrm{SNR}_{\mathrm{eff}}$ plot illustrates that, if $\mathrm{SNR}_{\mathrm{eff}}$ is the same regardless of which target it corresponds to, the BER will be approximately the same, especially at low σ_j/T cases. As a result, $\mathrm{SNR}_{\mathrm{eff}}$ can be used instead of BER as a criterion to compare the performance of different targets for a given input SNR. However, keep in mind that to achieve the same BER or $\mathrm{SNR}_{\mathrm{eff}}$, different targets may require different amounts of input SNR, as illustrated in Figure 57(b).

5.2.4 Summary

Research on perpendicular recording has been of increasing interest because of the potential for increased storage capacity as compared to longitudinal recording. Even though the same PRML detection process used in longitudinal recording can still be used for perpendicular recording, the target must be specifically designed for the perpendicular channel to obtain optimal performance.

A substantial performance improvement can be obtained by using the GPR target instead of the PR target, especially at high ND. At high ND, a longer target will provide a better match to the channel response than a shorter one because the effect of ISI becomes more severe at high ND. Irrespective of any media jitter noise level, the GPR target yields a better performance than the PR target. This is because the GPR target tends to whiten the noise at the input of the Viterbi detector. Because the GPR target is primarily a function of ND, SNR, and the media jitter noise amount, one needs to carefully design the GPR target for a given system condition so as to obtain a good performance.

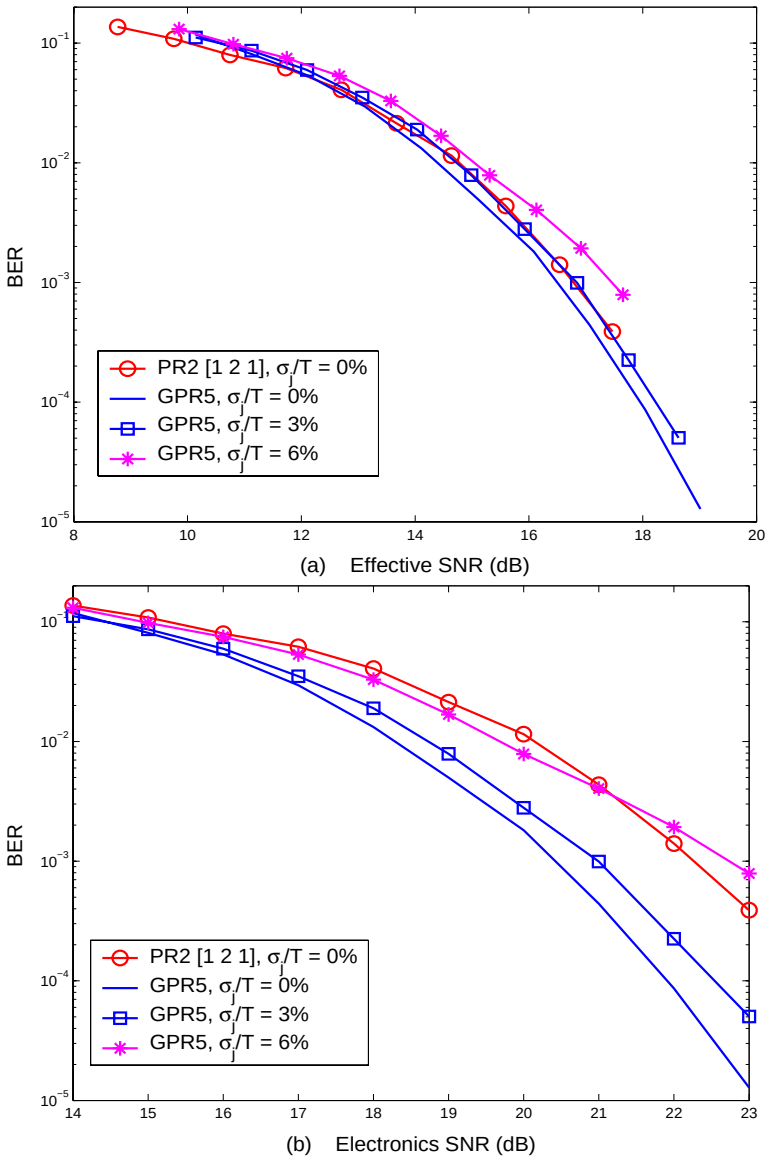


Figure 57: (a) BER vs. SNR_{eff} and (b) BER vs. Electronics SNR of different targets with various σ_j/T's at ND = 2.5.

We also showed that when σ_j/T is low, the effective distance compared with the Euclidean distance more accurately predicts which error sequence is most likely to occur in a system. Therefore, the performance of the Viterbi detector at high SNR can be determined by the error sequence that has the smallest effective distance. In addition, the resulting dominant error sequence is the same for all targets and different from longitudinal recording. Designing and using codes that avoid this error

sequence will further improve the system performance. Finally, we have demonstrated that $\mathrm{SNR}_{\mathrm{eff}}$ can be equivalently used instead of BER to measure the performance of different targets.

5.3 Timing Recovery for Fast Convergence

In practice, it is desirable for timing recovery to achieve synchronization as fast as possible. This means that all the initial phase and frequency offsets in a system during acquisition, and any phase and frequency changes during tracking should be recovered very quickly (i.e., within a fewer number of samples).

Today's magnetic recording read-channel chip architectures employ symbol-rate sampling and conventional timing recovery to reduce the system cost. In this configuration, which will be referred to as a *conventional receiver*, a T-spaced equalizer is used to shape the overall channel impulse response to the target response before performing ML equalization. However, we have shown in Section 2.4 that conventional timing recovery does not perform well if we want to recover all the sampling phase and frequency information very quickly, e.g., within 100 or 50 samples.

To improve the performance of conventional timing recovery, we exploit the idea of oversampling the received analog signal by twice the symbol rate to get more timing information. Because the oversampled system requires a fractionally-spaced equalizer instead of a T-spaced equalizer, it will also get all the benefits from a fractionally-spaced equalizer [63, 82]. For example, it is insensitive to a constant timing offset [63] in the system, as opposed to a T-spaced equalizer. With this idea, we propose the *oversampled PSP-based timing recovery* scheme [37] to achieve a fast convergence rate in the application of magnetic recording channels. Four system configurations are investigated and compared in this section, as shown in Figure 58.

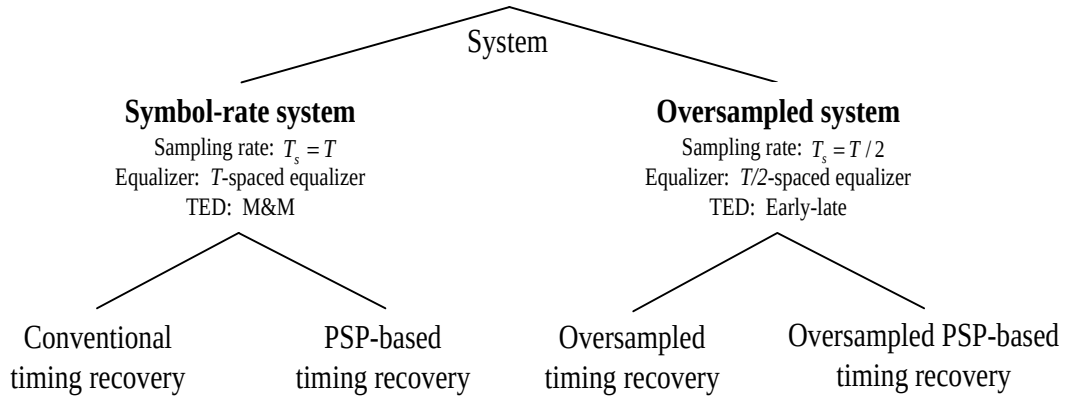


Figure 58: Diagram of different timing recovery schemes.

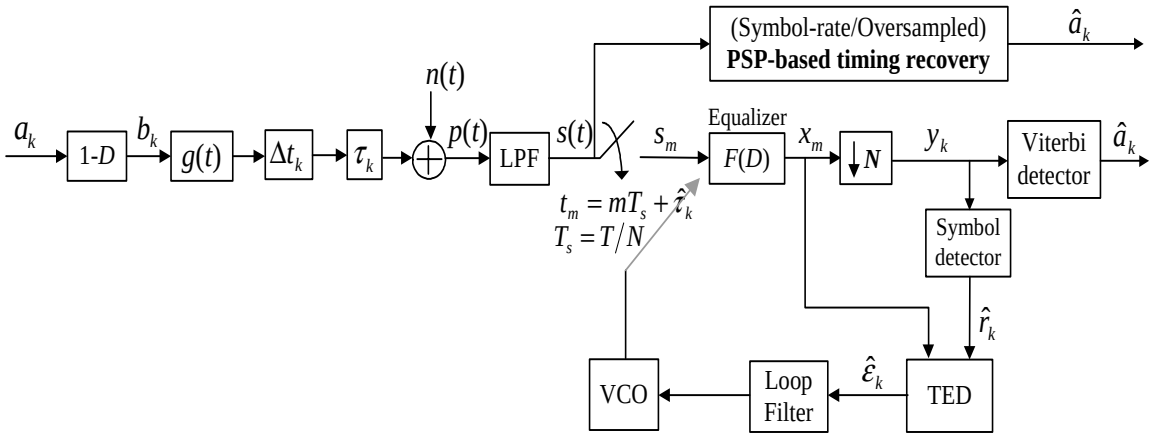


Figure 59: Magnetic recording channel model.

5.3.1 System Description

Consider the magnetic recording channel model shown in Figure 59, where the read-back signal can be written as [37, 38]

$$p(t) = \sum_{k=-\infty}^{\infty} a_k\{g(t - kT - \Delta t_k - \tau_k) - g(t - (k+1)T - \Delta t_{k+1} - \tau_k)\} + n(t). \quad (70)$$

The readback signal $p(t)$ is filtered by a seventh-order Butterworth low-pass filter, whose cutoff frequency is at $N/(2T)$, and is sampled at $t_m = mT_s + \hat{\tau}_k$ where $T_s = T/N$ is the sampling period, $N \in \{1, 2\}$ is an oversampling ratio, and $\hat{\tau}_k$ is the timing estimate of τ_k at time k ($k = \lfloor m/N \rfloor$ where $\lfloor \cdot \rfloor$ takes on the smallest integer value).

The T/N-spaced received sequence, s_m, is equalized by a T/N-spaced equalizer, $F(D)$, and then is downsampled to obtain a T-spaced sequence, y_k, (i.e., $y_k = x_{Nk}$) closely resembling a desired sequence, r_k. Then, TED utilizes x_m and $\hat{r}_k$ to generate the estimated timing error $\hat{\epsilon}_k$. The symbol-rate system (i.e., $N = 1$) uses the M&M TED [54] given in (4), i.e.,

$$\hat{\epsilon}_k = x(kT + \hat{\tau}_k)\hat{r}_{k-1} - x((k-1)T + \hat{\tau}_{k-1})\hat{r}_k. \tag{71}$$

Note that the constant K_T (as used in (4) to ensure that the S-curve slope of (71) is unity at the origin) will be included in the PLL gain parameters. For the oversampled system (i.e., $N = 2$) case, we consider the early-late TED [45], which is expressed as

$$\hat{\epsilon}_k = \hat{r}_k \left\{ x(kT + \frac{T}{2} + \hat{\tau}_k) - x(kT - \frac{T}{2} + \hat{\tau}_{k-1}) \right\}. \tag{72}$$

Then, the timing estimate is updated by a second-order PLL according to (5) – (6). Note that the symbol detector used in the timing loop is the Viterbi detector with a decision delay of $4T$.

5.3.2 Incorporating the Equalizer in PSP-Based Timing Recovery

To incorporate the equalizer in PSP-based timing recovery, let $s_i^{(p,q)_k}$ denote the sampler output at the sampling time index, i, associated with the survivor path leading to the state transition (p, q) at time k (or at the k-th stage). For example, as shown in Figure 14 with $N = 1$, $s_k^{(1,2)_k} = s(kT+\hat{\tau}_k(1))$. Similarly, $s_{k-1}^{(1,2)_k} = s((k-1)T+\hat{\tau}_{k-1}(0))$.

Let an M-tap T/N-spaced equalizer take the form $F(D) = \sum_{i=-K}^{K} f_i D^i$. Then, the equalizer output at symbol interval associated with (p, q) at time k can be expressed as

$$x_m^{(p,q)_k} = \sum_{i=-K}^{K} f_i s_{m-K-i}^{(p,q)_k} \tag{73}$$

Note that $y_k(p) = x_{Nk}^{(p,q)_k}$ and $y_{k+\frac{K}{N}}$ will correspond to a_k. Once the equalizer output is determined, the process of the timing update operation in PSP-based timing recovery is the same as explained in Section 3.3.

(C-1) Initialize $\Phi_0(p) = 0$ for $\forall p$

*(C-2) Initialize $\hat{\tau}_k(p) = 0$ and $\hat{\theta}_k(p) = 0$ for $k < K/N$ and $\forall p$

(C-3) For $k = 0, 1, \ldots, L + \nu - 1 + (K/N)$

(C-4) For $q = 0, 1, \ldots, Q - 1$

*(C-5) $s_{Nk+j}^{(p,q)_k} = s((Nk + j)T_s + \hat{\tau}_k(p))$ for $j = 0, \ldots, N - 1$ and $\forall p$

(C-6) $x_{Nk+j}^{(p,q)_k} = \sum_{i=-K}^{K} f_i s_{(Nk+j)-K-i}^{(p,q)_k}$ for $j = 0, \ldots, N - 1$ and $\forall p$

(C-7) $y_k(p) = x_{Nk}^{(p,q)_k}$ for $\forall p$

(C-8) $\rho_k(p, q) = |y_k(p) - \hat{r}(p, q)|^2$ for $\forall p$

(C-9) $\pi_{k+1}(q) = \arg\min_p\{\Phi_k(p) + \rho_k(p, q)\}$

(C-10) $\Phi_{k+1}(q) = \Phi_k(\pi_{k+1}(q)) + \rho_k(\pi_{k+1}(q), q)$

(C-11) $\mathbf{S}_{k+1}(q) = [\mathbf{S}_k(q) \mid \pi_{k+1}(q)]$

*(C-12) If $N = 1$ [M&M TED],

$\hat{\epsilon} = y_k(\pi_{k+1}(q))\hat{r}(\pi_k(\pi_{k+1}(q)), \pi_{k+1}(q)) - y_{k-1}(\pi_k(\pi_{k+1}(q)))\hat{r}(\pi_{k+1}(q), q)$

If $N = 2$ [Early-late TED],

$\hat{\epsilon} = \hat{r}(\pi_{k+1}(q), q)\left\{x_{Nk+1}^{(\pi_{k+1}(q),q)_k} - x_{Nk-1}^{(\pi_{k+1}(q),q)_k}\right\}$

*(C-13) $\hat{\theta}_{k+1}(q) = \hat{\theta}_k(\pi_{k+1}(q)) + \kappa\hat{\epsilon}$

*(C-14) $\hat{\tau}_{k+1}(q) = \hat{\tau}_k(\pi_{k+1}(q)) + \xi\hat{\epsilon} + \hat{\theta}_{k+1}(q)$

(C-15) End

(C-16) End

(C-17) Extract $\hat{\mathbf{a}}$ from the survivor path that minimizes $\Phi_{L+\nu}$

Figure 60: PSP-based timing recovery algorithm with a T/N-spaced equalizer, where the lines beginning with * are the additional steps beyond the conventional receiver.

Figure 60 illustrates the algorithm for PSP-based timing recovery for $N \in \{1, 2\}$, where the lines beginning with * are the additional steps beyond the conventional receiver. Note that an amount of K/N will represent the delay in bit period. For simplicity, we assume that K is divisible by N. It should be pointed out that only $\hat{\tau}_i$ and $\hat{a}_i$ for $i = (K/N), (K/N) + 1, \cdots, L - 1 + (K/N)$ will correspond to the input sequence a_k because of the delay introduced by a T/N-spaced equalizer.

Table 6: 5-tap GPR targets for different systems.

Channels and systems		5-tap GPR target $H(D)$
Longitudinal	Symbol-rate	[1 0.613 -0.478 -0.626 -0.291]
	Oversampled	[1 0.419 -0.441 -0.544 -0.268]
Perpendicular	Symbol-rate	[1 1.429 1.097 0.465 0.099]
	Oversampled	[1 1.421 1.076 0.451 0.097]

5.3.3 Numerical Results and Discussion

We consider ND = 2.5 for both longitudinal and perpendicular recording channels with $\sigma_j/T = 3\%$ media jitter noise, $\sigma_w/T = 0.5\%$ clock jitter noise, and 0.4% frequency offset. The 5-tap GPR target and a 21-tap equalizer were designed at the SNR required to achieve BER = 10^{-5}. Table 6 shows the 5-tap GPR targets designed for different system conditions.

A linearized model of second-order PLL [9] (as described in Section 2.3.2) is used to design ξ and κ, assuming that there is no noise in the system and the S-curve slope [9] is unity at the origin. The PLL gain parameters were designed to recover phase and frequency changes in C bit periods (the smaller the C, the faster the convergence rate) as presented in Section 2.3. Note that the PLL gain parameters strongly depend on the chosen target, the total delay (denoted as dT) in the timing loop, a given C, and a TED algorithm. Here, we consider the case where the same PLL gain parameters are used during both acquisition and tracking modes.

Four timing recovery schemes shown in Figure 58 are compared. Note that each scheme experiences different loop delays. Obviously, the total loop delays of conventional, oversampled, PSP-based, and oversampled PSP-based timing recovery are $14T$, $9T$, $10T$, and $5T$, respectively (as a T-spaced equalizer, a $T/2$-spaced equalizer, and a symbol detector introduce delays of $10T$, $5T$, and $4T$, respectively). Finally, each BER point was computed using as many data packets as needed to collect at least 1000 error bits. One data packet consists of a C-bit preamble ($4T$ pattern) and

Table 7: PLL gain parameters for longitudinal recording for the 5-tap GPR target.

Convergence rate (in bit periods)		Timing recovery schemes			
		Conventional $d = 14$	PSP-based $d = 10$	Oversampled $d = 9$	Oversampled PSP-based $d = 5$
$C = 256$	ξ	0.0027	0.0028	0.0043	0.0045
	κ	3.11e-5	3.24e-5	5.22e-5	5.44e-5
$C = 100$	ξ	0.0057	0.0062	0.0098	0.0107
	κ	1.43e-4	1.63e-4	2.65e-4	3.09e-4
$C = 50$	ξ	0.0087	0.0098	0.0158	0.0189
	κ	3.91e-4	4.74e-4	7.71e-4	9.88e-4

a 4096-bit input data sequence.

5.3.3.1 Longitudinal Recording

The PLL gain parameters for different timing recovery schemes are shown in Table 7. Figure 61 shows the BER performance of different schemes using ξ and κ designed for $C = 256$. With perfect timing, the oversampled system itself offers a large performance gain over the symbol-rate system. This suggests that the oversampled system should be employed in a longitudinal recording channel.

As depicted in Figure 61, for a given architecture (PSP-based or conventional), the oversampled system outperforms the symbol-rate system in all cases. However, for a given system (oversampled or symbol-rate), PSP-based timing recovery performs just slightly better than the conventional one. This is because they operate at the *optimal point*, where ξ and κ were designed to minimize the steady-state error in the timing loop (based on a linearized model), regardless of the convergence rate. The ξ and κ designed for $C = 256$ can be given as an example. Nevertheless, the performance gain becomes high when employing ξ and κ designed for small C (i.e., when operating in a system that requires fast convergence), as illustrated in Figure 62. Clearly, oversampled PSP-based timing recovery yields the best performance among other timing recovery schemes in all cases. This can be implied that oversampled

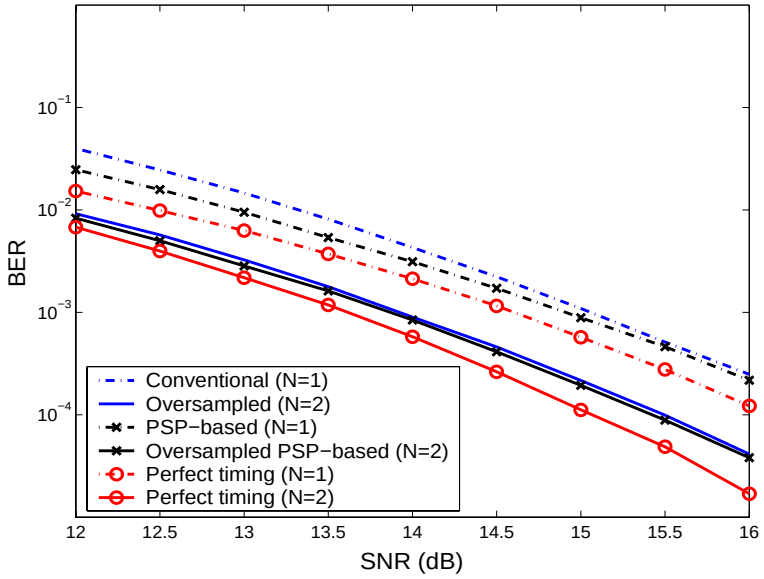


Figure 61: BER performance of different timing recovery schemes for longitudinal recording using ξ and κ designed for C=256.

Table 8: PLL gain parameters for perpendicular recording for the 5-tap GPR target.

Convergence rate (in bit periods)		Timing recovery schemes			
		Conventional $d = 14$	PSP-based $d = 10$	Oversampled $d = 9$	Oversampled PSP-based $d = 5$
$C = 100$	ξ	0.0070	0.0076	0.0129	0.0140
	κ	1.76e-4	2.02e-4	3.48e-4	4.06e-4
$C = 50$	ξ	0.0107	0.0121	0.0207	0.0248
	κ	4.83e-4	5.86e-4	1.01e-3	1.30e-3

PSP-based timing recovery can achieve faster convergence than any other scheme.

5.3.3.2 Perpendicular Recording

The PLL gain parameters for different timing recovery schemes are shown in Table 8. Unlike longitudinal recording, we observed that there is no significant performance gain between the oversampled system and the symbol-rate system in perpendicular recording when operating a system using ξ and κ designed for $C = 256$. However, a relatively large gain can be obtained between the oversampled system and the

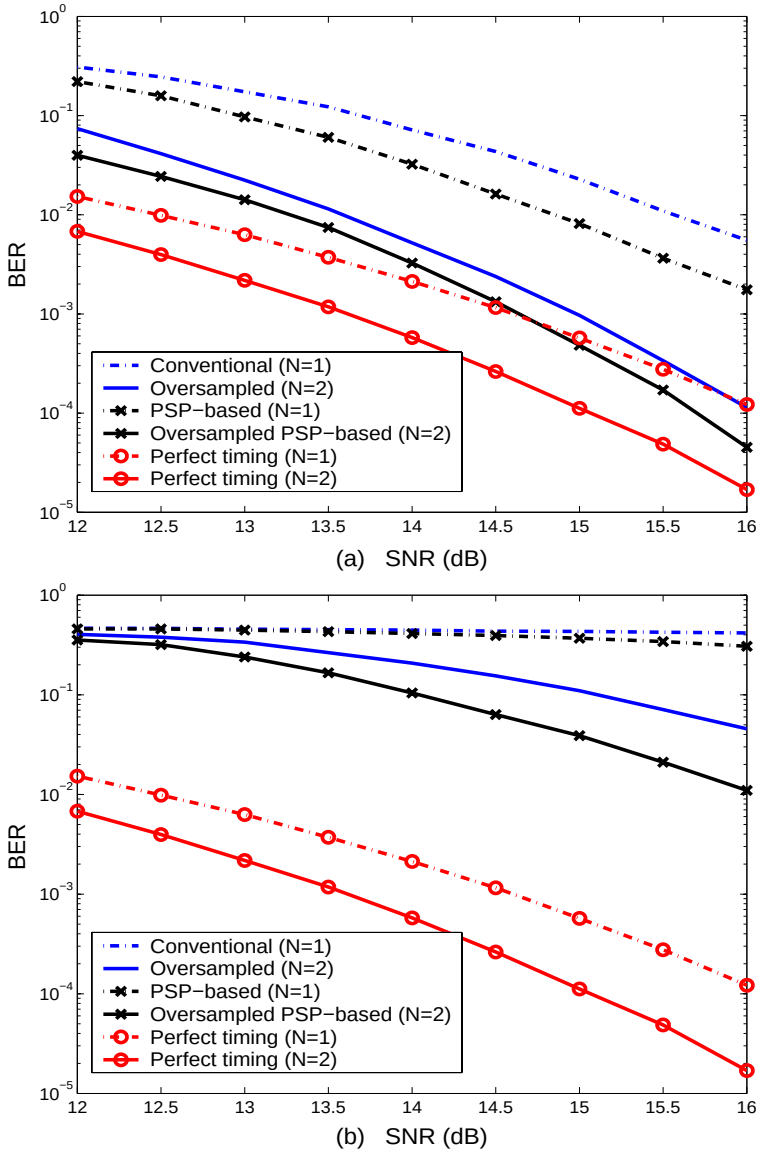


Figure 62: BER performance of different timing recovery schemes for longitudinal recording using ξ and κ designed for (a) $C = 100$ and (b) $C = 50$.

symbol-rate system, and between the PSP-based timing recovery architecture and the conventional timing recovery architecture when employing ξ and κ designed for $C = 100$ and $C = 50$, as depicted in Figure 63. Again, the oversampled PSP-based timing recovery scheme performs better than other schemes in all cases.

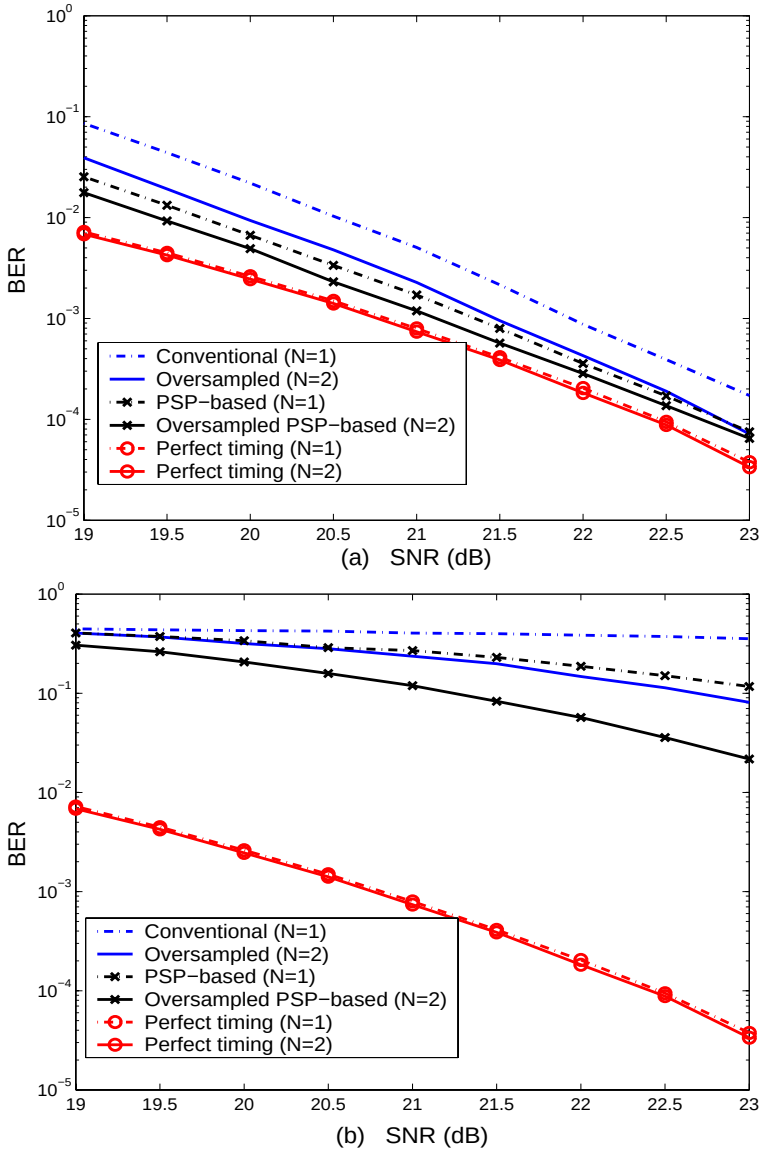


Figure 63: BER performance of different timing recovery schemes for perpendicular recording using ξ and κ designed for (a) $C = 100$ and (b) $C = 50$.

5.3.4 Summary

We investigated the idea of oversampling the received analog signal by twice the symbol rate to get more timing information in both longitudinal and perpendicular recording channels. With the oversampled method, we proposed the oversampled PSP-based timing recovery scheme to achieve fast convergence. Simulation results indicated that the oversampled PSP-based timing recovery scheme performs better

than other schemes, especially when operating in a system that requires fast convergence (i.e., when using the PLL gain parameters designed for small C).

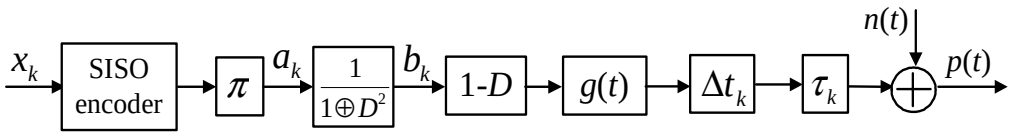
Although the oversampled system provides a better performance than the symbol-rate system in magnetic recording channels, one still needs to consider the implementation cost. For example, an analog-to-digital converter (i.e., a sampler) operating at twice symbol-rate sampling is very costly because of a very high data rate used in a hard disk drive. In addition, the complexity of PSP-based timing recovery is higher than conventional timing recovery. Therefore, all advantages gained by the oversampled PSP-based timing recovery scheme need to be balanced against increased implementation costs.

5.4 Iterative Timing Recovery

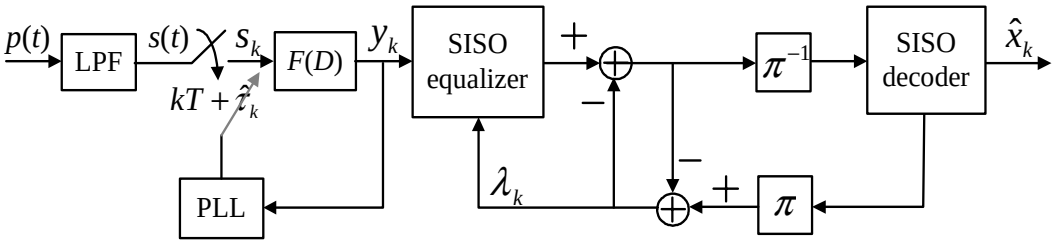
A large coding gain of iterative ECCs allows reliable operation at SNR lower than ever before. In magnetic recording systems, lower SNR not only reduces the cost of operation but also allows for higher storage capacity. In this section, we will investigate the performance of per-survivor iterative timing recovery in magnetic recording channels operating at low SNR. This experiment will help determine whether or not it is feasible to use this scheme in real-life applications, if compared to the conventional schemes used in today's magnetic recording read-channel chip architectures.

5.4.1 System Description

Consider a coded magnetic recording channel shown in Figure 64, where $g(t)$ is the transition response given in (53) for longitudinal recording and in (54) for perpendicular recording. The training bits will be "inserted" in a sequence b_k before passing it through the channel. At the receiver, after performing timing recovery, the training bits will be "discarded" at the equalizer output before feeding the resulting sequence into the turbo equalizer. We also assume that there is no frequency offset left in the system after the first iteration. This means that, when using the NBM scheme and



(a) Data encoding with a magnetic recording channel



(b) Conventional receiver

Figure 64: A magnetic recording channel model with a conventional receiver.

per-survivor iterative timing recovery, a first-order PLL will be employed to refine the samples after the first iteration.

5.4.2 Numerical Results and Discussion

We consider ND = 2 for both longitudinal and perpendicular recording channels with $\sigma_j/T = 3\%$ media jitter noise, $\sigma_w/T = 0.5\%$ clock jitter noise, and 0.4% frequency offset. The 3-tap GPR target and a 21-tap equalizer were designed at the SNR required to achieve BER = 10^{-5}. Again, the PLL gain parameters were designed for $C = 256$ based on a linearized model of PLL. Here, we again consider the case where the same PLL gain parameters are employed during both acquisition and tracking modes.

5.4.2.1 BER Performance

For longitudinal recording, the 3-tap GPR target is $H(D) = 1 + 0.098D - 0.702D^2$. The PLL gain parameters, which already includes a constant K_T as introduced in (4), for this channel are given in Table 9. Figure 65(a) compares the BER performance of different iterative timing recovery schemes. It is clear that per-survivor iterative

Table 9: PLL gain parameters for the 3-tap GPR target for different system conditions.

Channels		Receiver architectures	
		Conventional receiver & NBM scheme	Per-survivor iterative timing recovery
Longitudinal	ξ	0.00254	0.00263
	κ	2.93e-5	3.06e-5
Perpendicular	ξ	0.00398	0.00413
	κ	4.60e-5	4.81e-5

timing recovery outperforms the conventional receiver. Also, it performs better than the NBM scheme, especially at high SNR. Specifically, it can provide more than a 2 dB gain over the NBM scheme with 10 iterations at BER = 10^{-4}. Note that there is a big performance gap between per-survivor iterative timing recovery and the system with perfect timing, especially at high SNR. This might be because timing recovery suffers from a frequency offset component present in the system.

Similarly, for perpendicular recording, the 3-tap GPR target is $H(D) = 1 + 1.148D + 0.475D^2$. The PLL gain parameters for this channel are also given in Table 9. Figure 65(b) compares the BER performance of different schemes. Again, per-survivor iterative timing recovery performs better than the conventional receiver and the NBM scheme, especially at high SNR. Specifically, it can provide almost a 1 dB gain over the NBM scheme with 10 iterations at BER = 10^{-4}.

5.4.2.2 Complexity Versus Performance

It is worth comparing the performance of different iterative timing recovery schemes when they approximately have the same complexity (i.e., the same number of operations), as discussed in Section 4.6.1. With an equalizer in a system and the information given in Tables 1 and 2 (in Section 4.5.3), we can count the total number of operations of each scheme, including the SISO decoder, as illustrated in Table 10,

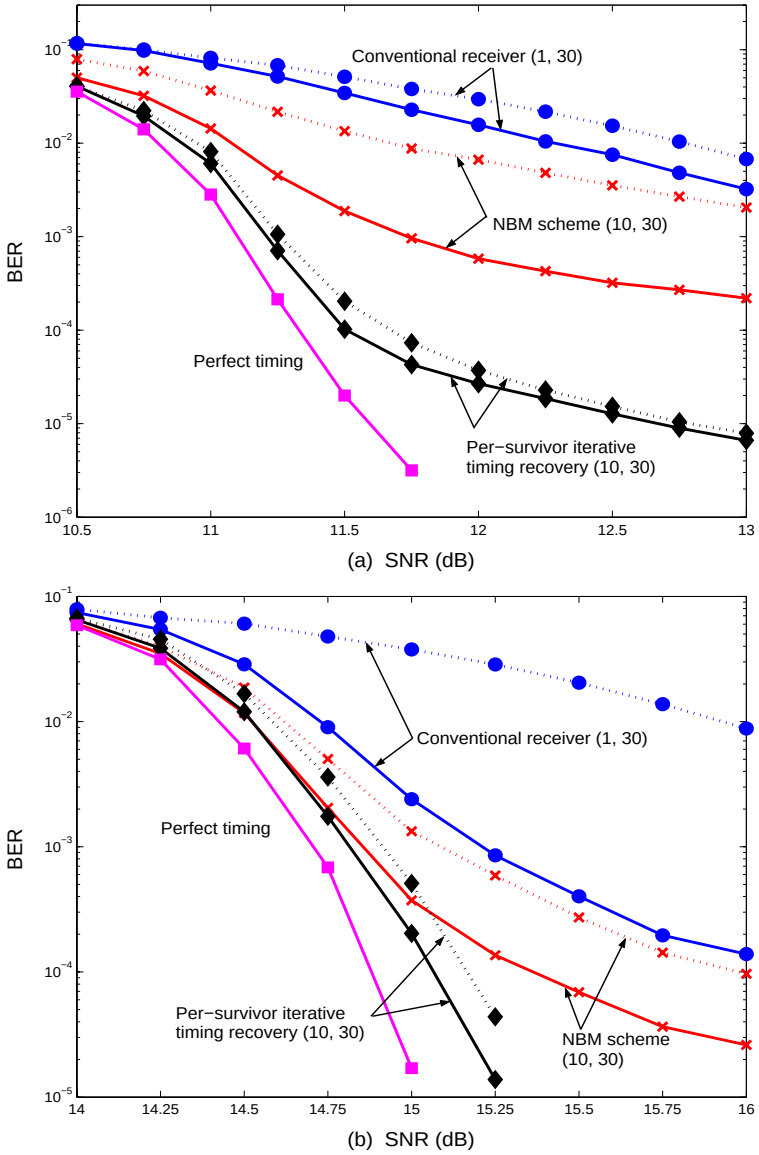


Figure 65: BER performance of different iterative timing recovery schemes for (a) longitudinal recording and (b) perpendicular recording at ND = 2, $\sigma_w/T = 0.5\%$, $\sigma_j/T = 3\%$, and 0.4% frequency offset.

where N is the number of iterations. Note that equalization using an N_{eq}-tap equalizer requires $N_{eq} - 1$ additions and N_{eq} multiplications (where $N_{eq} = 21$ is used in this work), and per-survivor iterative timing recovery performs equalization at each trellis state during forward and backward recursions.

Table 10: The total number of operations of each iterative timing recovery scheme used in magnetic recording systems.

Schemes	Number of operations (per bit)		
	Addition	Multiplication	Total
Conventional receiver	$106 + 330N$	$48 + 595N$	$154 + 925N$
NBM scheme	$436N$	$643N$	$1079N$
Per-survivor iterative	$1170N$	$955N$	$2125N$
Perfect timing	$103 + 330N$	$42 + 595N$	$145 + 925N$

Again, we consider the total number of operations when comparing the performance of iterative timing recovery schemes. We first assume that the current technology can support the total number of operations equal to 4 iterations of per-survivor iterative timing recovery. Then, as given in Table 10, it can be shown that per-survivor iterative timing recovery with 4 iterations has the total number of operations approximately equal to the conventional receiver with 10 iterations, the NBM scheme with 8 iterations, and the system with perfect timing with 10 iterations.

Figure 66 compares the performance of different iterative timing recovery schemes when they approximately have the same complexity. For longitudinal recording, it is evident that per-survivor iterative timing recovery performs much better than the conventional receiver and the NBM scheme, especially at high SNR. For perpendicular recording, although per-survivor iterative timing recovery performs slightly worse than the NBM scheme at low SNR, it outperforms the NBM scheme at high SNR. This implies that, when SNR is high enough, per-survivor iterative timing recovery can perform quite well even with a fewer iterations. Therefore, at low to moderate complexity, per-survivor iterative timing recovery still performs better than the conventional schemes, especially at high SNR. This is because per-survivor iterative timing recovery can correct a cycle slip much more efficiently than other schemes, especially at high SNR.

Finally, it is also worth comparing the performance of iterative timing recovery

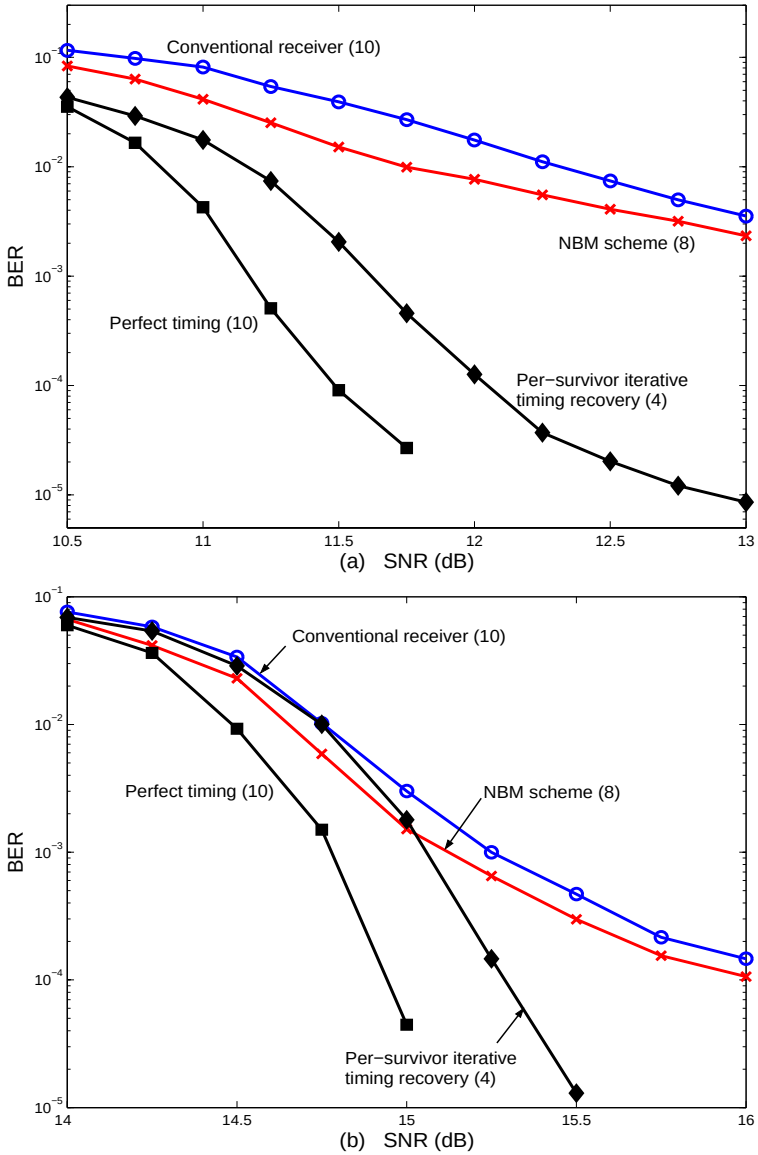


Figure 66: BER performance of different iterative timing recovery schemes with the same complexity for (a) longitudinal recording and (b) perpendicular recording channels at ND = 2 with $\sigma_w/T = 0.5\%$, $\sigma_j/T = 0.3\%$, and 0.4% frequency offset.

schemes when they approximately have the same complexity at different NDs. This experiment will tell us how much ND improvement we can obtain when all schemes have the same complexity. In this experiment, the 3-tap GPR target is used for all NDs. The target and its corresponding equalizer were designed at the SNR required to achieve BER = 10^{-5} for each ND. Figure 67 plots the SNR required to achieve

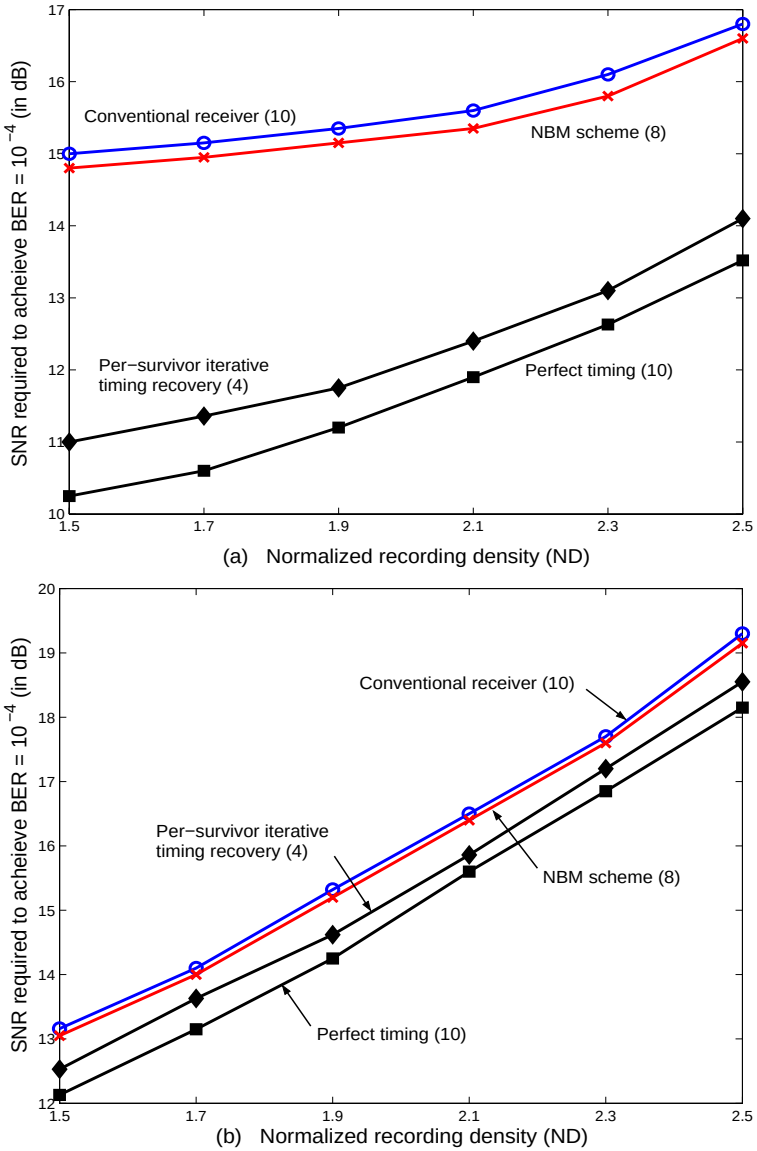


Figure 67: SNR required to achieve BER $= 10^{-4}$ (in dB) versus ND of different iterative timing recovery schemes with the same complexity for (a) longitudinal recording and (b) perpendicular recording channels with $\sigma_w/T = 0.5\%$, $\sigma_j/T = 0.3\%$, and 0.4% frequency offset.

BER $= 10^{-4}$ as a function of NDs. It is apparent that, for longitudinal recording, a large ND improvement gain can be obtained from per-survivor iterative timing recovery if compared to the conventional receiver and the NBM scheme. Nevertheless, only a small ND improvement gain is obtained from per-survivor iterative timing

recovery when operating in perpendicular recording channels. This suggests that per-survivor iterative timing recovery is of more worth when it is employed in longitudinal recording channels than in perpendicular recording channels.

5.4.3 Summary

We investigated the performance of different iterative timing recovery schemes in magnetic recording channels (both longitudinal and perpendicular recording) operating at low SNR. It has been shown that per-survivor iterative timing recovery performs better than conventional schemes when operating in a moderate system condition (e.g., with ND = 2, $\sigma_w/T = 0.5\%$ clock jitter noise, $\sigma_j/T = 3\%$ media jitter noise, and 0.4% frequency offset). Additionally, we have shown that per-survivor iterative timing recovery can also achieve higher NDs than other iterative timing recovery schemes, especially in longitudinal recording channels. Therefore, per-survivor iterative timing recovery is worth being employed in magnetic recording systems.

CHAPTER 6

CONCLUSION

Timing recovery is the process of synchronizing the sampler with the received analog signal. Sampling at the wrong times can have a devastating impact on overall performance. Improving the performance of timing recovery will give rise to improved reliability of an entire system. In this work, we developed and investigated new timing recovery schemes that perform better than conventional ones for the systems with and without ECCs.

6.1 Summary

In Chapter 2, a brief overview of conventional timing recovery that is based on a PLL was given. A method of designing the PLL gain parameters based on a linearized model of PLL was introduced. We have shown that for low to moderate SNRs, conventional timing recovery does not perform well, especially when the timing error is large or when operating in a system that requires fast convergence.

Chapter 3 dealt with the problem of timing recovery in the absence of ECCs (i.e., in uncoded systems). We proposed a PSP-based timing recovery scheme to jointly perform timing recovery and equalization. It has been shown that PSP-based timing recovery performs better than conventional timing recovery, especially when the timing jitter is severe, and also achieves faster convergence than conventional timing recovery. We also investigated different approaches to reduce the complexity of PSP-based timing recovery. We found that the M- and T- algorithms can be used to reduce the complexity of PSP-based timing recovery with acceptable performance if their parameters are chosen suitably.

Chapter 4 dealt with the problem of timing recovery in the presence of ECCs (i.e., in coded systems). We proposed a per-survivor iterative timing recovery scheme to jointly perform timing recovery, equalization, and error-correction decoding. It has been shown that per-survivor iterative timing recovery outperforms other iterative timing recovery schemes, especially when the timing error is large. This is because per-survivor iterative timing recovery can automatically correct a cycle slip much more efficiently than the others. A reduced-complexity version of per-survivor iterative timing recovery is also investigated. Again, we found that the M- and T- algorithms can be used to reduce the complexity of per-survivor iterative timing recovery with acceptable performance if their parameters are suitably chosen. Furthermore, we also proposed to use the EXIT chart as a tool to compare and predict the performance of iterative timing recovery schemes. We have showed that the EXIT chart can be equivalently used instead of BER as a measure to compare the performance of different iterative timing recovery schemes, assuming that there is no cycle slip. Specifically, the system performance predicted by the EXIT chart coincides with that obtained by simulating data transmission over the complete iterative receiver, especially when the coded block length is large.

In Chapter 5, we first proposed the suitable GPR targets for a perpendicular recording channel and found that the dominant error sequence for this channel is {+2,-2}, as opposed to {+2,-2, +2} for a longitudinal recording channel. Then, we demonstrated that the effective SNR can be equivalently used instead of BER to measure the performance of different targets, considering that the BER computation takes a considerable amount of simulation time. We also investigated the idea of oversampling the received analog signal by twice the symbol rate to get more timing information in magnetic recording channels. With this idea, we proposed the oversampled PSP-based timing recovery scheme to achieve fast convergence for magnetic recording channels (both longitudinal and perpendicular recording). It has been shown that the

oversampled PSP-based timing recovery scheme performs better than other schemes, especially when operating in a system that requires fast convergence. Finally, we investigated the performance of iterative timing recovery schemes in magnetic recording channels operating at low SNR. Simulation results have shown that per-survivor iterative timing recovery performs better and achieves higher ND than other iterative timing recovery schemes. Therefore, per-survivor iterative timing recovery is worth being employed in magnetic recording systems, if compared to conventional schemes used in today's magnetic recording read-channel chip architectures.

6.2 Future Work

In this work, we proposed new timing recovery schemes for the systems with and without ECCs. It has been shown that the performance of the proposed schemes is quite promising if compared to conventional schemes. Nonetheless, most results are based on simulations. To fully understand their architectures, additional analytical expressions (e.g., a lower bound, a predicted BER, etc.) would be of great interest.

Performance analysis of iterative timing recovery schemes is difficult to determine because of their complexity. Although we proposed to use the EXIT chart as a tool to compare and predict their performances, this method is still based mostly on simulations. Therefore, it would be beneficial to have a purely theoretical tool for measuring and predicting their performance.

Only the T- and M- algorithms are considered in this work as a means to reduce the complexity of the proposed timing recovery schemes. It would also be interesting to investigate other possibilities to reduce their complexity with acceptable performance. For example, we could incorporate the constraint codes, e.g., run-length limited (RLL) codes [32], in the system. The advantage of this technique is not only to reduce the number of branches in the trellis but also help facilitate timing recovery. However, it has a drawback of lowering the code rate. A fair performance

comparison of this approach against the system without a constraint code would be of interest. Additionally, many works have proposed to use a soft-output Viterbi algorithm (SOVA) [28] in iterative detection [28, 29]. Therefore, it might be a good idea to embed the timing recovery step inside SOVA based on PSP so as to perform timing recovery and equalization jointly, for which we will denote this technique as "PSP-SOVA." We can then use PSP-SOVA in place of PSP-BCJR to reduce the complexity of per-survivor iterative timing recovery. The performance comparison of per-survivor iterative timing recovery using PSP-BCJR and PSP-SOVA, for given complexity, would be interesting.

Finally, in magnetic recording channels, the noise seen at the equalizer output is normally colored noise, especially at high normalized recording densities. To cope with the colored noise, the technique known as *noise-predictive maximum-likelihood* (NPML) [20, 62] has been introduced. It would be of interest to incorporate this technique in PSP-based timing recovery and per-survivor iterative timing recovery and then investigate their performance. We expect a large performance improvement by using this technique, especially at high normalized recording densities. Furthermore, the noise in magnetic recording channels also possesses another unique feature, which is *data dependent* [51]. Media jitter noise can be given as an example of this type of noise. This media jitter noise depends on the data pattern written on the disk and can contribute a significant portion of the total noise. The so-called pattern-dependent noise-predictive (PDNP) technique [51] has been proposed to combat the pattern dependence of media noise. Therefore, it would also be interesting to incorporate the PDNP technique in PSP-based timing recovery and per-survivor iterative timing recovery and investigate their performance. We again expect a large performance improvement by using this technique, especially when operating in a channel at high normalized recording densities or at high media jitter noise levels.

APPENDIX A

DERIVATION OF THE STARTING STATE ASSOCIATED WITH THE BEST STATE TRANSITION

This appendix explains how to derive the expression given in (B-8) that is used to choose the *best* (forward) state transition for PSP-BCJR. Following the notations in [5], [8], the starting state associated with the best state transition leading to state q at time $k+1$ is chosen according to

$$\hat{p} = \arg\max_{p} \left\{ \Pr[\Psi_k = p, \Psi_{k+1} = q | y_k, \mathbf{y}_{l<k}] \right\}, \tag{74}$$

where Ψ_k is the trellis state at time k, y_k is the k-th channel observation, and $\mathbf{y}_{l<k}$ a collection of $\{y_l\}$ for $l < k$.

Using Bayes' rule, the probability in (74) can be rewritten as

$$\begin{aligned} \Pr[\Psi_k = p, \Psi_{k+1} = q | y_k, \mathbf{y}_{l<k}] &= \frac{\Pr[\Psi_k = p, \Psi_{k+1} = q, y_k, \mathbf{y}_{l<k}]}{\Pr[y_k, \mathbf{y}_{l<k}]} \\ &= \frac{\Pr[\Psi_{k+1} = q, y_k | \Psi_k = p, \mathbf{y}_{l<k}] \cdot \Pr[\Psi_k = p, \mathbf{y}_{l<k}]}{\Pr[y_k, \mathbf{y}_{l<k}]} \\ &= \frac{\gamma_k(p,q) \cdot \alpha_k(p)}{\sum_u \alpha_{k+1}(u)}, \end{aligned} \tag{75}$$

where

$$\begin{aligned} \Pr[\Psi_{k+1} = q, y_k | \Psi_k = p, \mathbf{y}_{l<k}] &= \Pr[\Psi_{k+1} = q, y_k | \Psi_k = p] \\ &= \gamma_k(p,q) \\ \Pr[\Psi_k = p, \mathbf{y}_{l<k}] &= \alpha_k(p) \\ \Pr[y_k, \mathbf{y}_{l<k}] &= \sum_u \Pr[\mathbf{y}_{l<k+1}, \Psi_{k+1} = u] \end{aligned}$$

$$= \sum_{u} \alpha_{k+1}(u).$$

Substituting (75) in (74), one obtains

$$\hat{p} = \arg\max_{p}\{\gamma_k(p,q) \cdot \alpha_k(p)\} \tag{76}$$

by ignoring the term irrelevant to maximization.

APPENDIX B

DERIVATION OF THE STARTING STATE ASSOCIATED WITH THE BEST BACKWARD STATE TRANSITION

In this appendix, we derive the expression given in (B-21) that is used to select the *best* backward state transition in PSP-BCJR, which can be explained as follows. Following the notations in [5], [8], the starting state associated with the best backward state transition leading to state p at time k is chosen according to

$$\hat{q} = \arg\max_{q} \left\{ \Pr[\Psi_k = p, \Psi_{k+1} = q | y_k, \mathbf{y}_{l>k}] \right\}, \tag{77}$$

where $\mathbf{y}_{l>k}$ a collection of $\{y_l\}$ for $l > k$.

Using Bayes' rule, the probability in (77) can be rewritten as

$$\begin{aligned} \Pr[\Psi_k = p, \Psi_{k+1} = q | y_k, \mathbf{y}_{l>k}] &= \frac{\Pr[\Psi_k = p, \Psi_{k+1} = q, y_k, \mathbf{y}_{l>k}]}{\Pr[y_k, \mathbf{y}_{l>k}]} \\ &= \frac{\Pr[\Psi_k = p, y_k | \Psi_{k+1} = q, \mathbf{y}_{l>k}] \cdot \Pr[\Psi_{k+1} = q, \mathbf{y}_{l>k}]}{\Pr[y_k, \mathbf{y}_{l>k}]} \\ &= \frac{\gamma_k(p,q) \cdot \beta_{k+1}(q) \cdot \Pr[\Psi_k = p]}{\sum_u \beta_k(u) \Pr[\Psi_k = u]}, \end{aligned} \tag{78}$$

where

$$\begin{aligned} \Pr[\Psi_k = p, y_k | \Psi_{k+1} = q, \mathbf{y}_{l>k}] &= \Pr[\Psi_k = p, y_k | \Psi_{k+1} = q] \\ &= \Pr[\Psi_k = p, y_k, \Psi_{k+1} = q] / \Pr[\Psi_{k+1} = q] \\ &= \Pr[\Psi_{k+1} = q, y_k | \Psi_k = p] \cdot \Pr[\Psi_k = p] / \Pr[\Psi_{k+1} = q] \\ &= \gamma_k(p,q) \cdot \Pr[\Psi_k = p] / \Pr[\Psi_{k+1} = q] \\ \Pr[\Psi_{k+1} = q, \mathbf{y}_{l>k}] &= \Pr[\mathbf{y}_{l>k} | \Psi_{k+1} = q] \cdot \Pr[\Psi_{k+1} = q] \end{aligned}$$

$$
\begin{aligned}
&= \beta_{k+1}(q) \cdot \Pr[\Psi_{k+1} = q] \\
\Pr[y_k, \mathbf{y}_{l>k}] &= \sum_u \Pr[\mathbf{y}_{l>k-1}, \Psi_k = u] \\
&= \sum_u \Pr[\mathbf{y}_{l>k-1} | \Psi_k = u] \Pr[\Psi_k = u] \\
&= \sum_u \beta_k(u) \Pr[\Psi_k = u].
\end{aligned}
$$

Substituting (78) in (77), one obtains

$$\hat{q} = \arg\max_q \{\gamma_k(p, q) \cdot \beta_{k+1}(q)\} \tag{79}$$

by ignoring the terms irrelevant to maximization and assuming that all next states are equally likely.

REFERENCES

[1] Anastasopoulos, A., *Adaptive soft-in soft-output algorithms for iterative detection*. PhD thesis, University of Southern California, California, August 1999.

[2] Anastasopoulos, A. and Chugg, K. M., "Adaptive soft-in soft-output algorithms for iterative detection with parametric uncertainty," *IEEE Trans. Commun.*, vol. 48, pp. 1638–1649, October 2000.

[3] Anastasopoulos, A. and Chugg, K. M., "Adaptive iterative detection for phase tracking in turbo-coded systems," *IEEE Trans. Commun.*, vol. 49, pp. 2135–2144, December 2001.

[4] Andrea, A. N., Mengali, U., and Vitetta, G. M., "Approximate ML decoding of coded PSK with no explicit carrier phase reference," *IEEE Trans. Commun.*, vol. 42, pp. 1033–1039, Feb/Mar/Apr 1994.

[5] Bahl, L. R., Cocke, J., Jelinek, F., and Raviv, J., "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inform. Theory*, vol. IT-20, pp. 248–287, March 1974.

[6] Barbosa, L. C., "Maximum likelihood sequence estimators: A geometric view," *IEEE Trans. Inform. Theory*, vol. 35, pp. 419–427, March 1989.

[7] Barry, J. R., Kavčić, A., McLaughlin, S. W., and Nayak, A. R., "Iterative timing recovery," *IEEE Signal Processing Magazine*, vol. 21, pp. 89–102, January 2004.

[8] Barry, J. R., Lee, E. A., and Messerschmitt, D. G., *Digital Communication*. Boston, Massachusetts: Kluwer Academic Publishers, third ed., 2003.

[9] Bergmans, J. W., *Digital baseband transmission and recording*. Boston, Massachusetts: Kluwer Academic Publishers, 1996.

[10] Berrou, C., Glavieux, A., and Thitimajshima, P., "Near Shannon limit error-correcting coding and decoding: Turbo codes," in *Proc. of ICC'93*, vol. 2, pp. 1064–1070, May 1993.

[11] Burr, A. G. and Zhang, L., "Iterative joint synchronisation and turbo-decoding," *IEEE International Symposium on Information Theory*, p. 414, June 30 – July 5 2002.

[12] Caroselli, J. and Wolf, J. K., "Error event characterization of partial response systems in magnetic recording systems with medium noise," in *Proc. of Globecom'98*, vol. 5, pp. 2724–2728, November 1998.

[13] Chiavaccini, E. and Vitetta, G. M., "A per-survivor phase-estimation algorithm for detection of PSK signals," *IEEE Trans. Commun.*, vol. 49, pp. 2059–2061, December 2001.

[14] Chugg, K. M., Anastasopoulos, A., and Chen, X., *Iterative Detection - Adaptivity, Complexity Reduction, and Applications.* Boston, Massachusetts: Kluwer Academic Publishers, 2000.

[15] Cideciyan, R. D., Dolivo, F., Hermann, R., Hirt, W., and Schott, W., "A PRML system for digital magnetic recording," *IEEE J. Selected Areas Commun.*, vol. 10, pp. 38–56, January 1992.

[16] Colavolpe, G., Ferrari, G., and Raheli, R., "Reduced-state BCJR-type algorithms," *IEEE J. Selected Areas Commun.*, vol. 19, pp. 848–859, May 2001.

[17] Conway, T., "A new target response with parity coding for high density magnetic recording channels," *IEEE Trans. Magnetics*, vol. 34, pp. 2382–2386, June 1998.

[18] Corazza, G. E., Polydoros, A., Coralli, A. V., Salmi, P., and Cioni, S., "Performance analysis of PSP based joint data decoding and phase/frequency estimation in satellite communications," in *Proc. IEEE 11th Int. Symp. Personal, Indoor and Mobile Radio Communication, PIMRC2000*, vol. 1, pp. 544–548, September 2000.

[19] Duel-Hallen, A. and Heegard, C., "Delayed decision-feedback sequence estimation," *IEEE Trans. Commun.*, vol. 37, pp. 428–436, May 1989.

[20] Eleftheriou, E. and Hirt, W., "Noise-predictive maximum-likelihood (NPML) detection for the magnetic recording channel," in *Proc. of ICC'96*, vol. 1, pp. 556–560, June 1996.

[21] Esteves, E. S. and Sampaio-Neto, R., "A per-survivor phase acquisition and tracking algorithm for detection of TCM signals with phase jitter and frequency error," *IEEE Trans. Commun.*, vol. 45, pp. 1381–1384, November 1997.

[22] Eyuboğlu, M. V. and Qureshi, S. U., "Reduced-state sequence estimation for coded modulation on intersymbol interference channels," *IEEE J. Selected Areas Commun.*, vol. 7, pp. 989–995, August 1989.

[23] Fitzpatrick, J., Wolf, J. K., and Barbosa, L., "New equalizer targets for sampled magnetic recording system," in *Proc. of the 25th Asilomar Conference on Signals Systems and Computers*, vol. 1, pp. 30–34, November 1991.

[24] Forney, G. D., "Maximum-likelihood sequence estimation of digital sequences in the presence of intersymbol interference," *IEEE Trans. Inform. Theory*, vol. IT-18, pp. 363–378, May 1972.

[25] Franz, V. and Anderson, J. B., "Concatenated decoding with a reduced-search BCJR algorithm," *IEEE J. Selected Areas Commun.*, vol. 16, pp. 186–195, February 1998.

[26] Gallager, R., "Low-density parity-check codes," *IRE Trans. Inform. Theory*, vol. IT-8, pp. 21–28, January 1962.

[27] Georghiades, C. N. and Snyder, D. L., "The expectation-maximization algorithm for symbol unsynchronized sequence detection," *IEEE Trans. Commun.*, vol. 39, pp. 54–61, January 1991.

[28] Hagenauer, J. and Hoeher, P., "A Viterbi algorithms with soft-decision outputs and its applications," in *Proc. of Globecom'89*, pp. 1680–1686, November 1989.

[29] Hagenauer, J., Offer, E., and Papke, L., "Iterative decoding of binary block and convolutional codes," *IEEE Trans. Inform. Theory*, vol. 42, pp. 429–445, March 1996.

[30] Heegard, C. and Wicker, S. B., *Turbo Coding.* Boston, Massachusetts: Kluwer Academic Publishers, 1999.

[31] Iltis, R. A., "A Bayesian maximum-likelihood sequence estimation algorithm for a *priori* unknown channels and symbol timing," *IEEE J. Selected Areas Commun.*, vol. 10, pp. 579–588, April 1992.

[32] Immink, K. A. S., "Runlength-limited sequences," *Proceeding of the IEEE*, vol. 78, pp. 1745–1759, November 1990.

[33] Jin, X. and Kavčić, A., "Cycle-slip detection using soft-output information," in *Proc. of ICC'01*, vol. 9, pp. 2706–2710, June 2001.

[34] Kovintavewat, P. and Barry, J. R., "EXIT chart analysis for iterative timing recovery," to appear in *Proc. of Globecom'04,* Dallas, Texas, November 29 – December 3, 2004.

[35] Kovintavewat, P., Barry, J. R., Erden, M. F., and Kurtas, E., "Per-survivor iterative timing recovery for coded partial response channels," to appear in *Proc. of Globecom'04,* Dallas, Texas, November 29 – December 3, 2004.

[36] Kovintavewat, P., Barry, J. R., Erden, M. F., and Kurtas, E., "Per-survivor processing (PSP) -based timing recovery for uncoded partial response channels," to appear in *Proc. of ICC'04,* Paris, France, June 20-24, 2004.

[37] Kovintavewat, P., Erden, M. F., Kurtas, E., and Barry, J. R., "A new timing recovery architecture for fast convergence," *IEEE International Symposium on Circuits and Systems (ISCAS)*, vol. 2, pp. 13–16, May 2003.

[38] Kovintavewat, P., Erden, M. F., Kurtas, E., and Barry, J. R., "Oversampled timing recovery for magnetic recording channels," in *Proc. of the IEEE International Conference on Magnetics (Intermag) 2003*, pp. DT–06, March 30 – April 3, 2003.

[39] Kovintavewat, P., Ozgunes, I., Kurtas, E., Barry, J. R., and McLaughlin, S. W., "Generalized partial response targets for perpendicular recording with jitter noise," *IEEE Trans. Magnetics*, vol. 38, pp. 2340–2342, September 2002.

[40] Kovintavewat, P., Ozgunes, I., Kurtas, E., Barry, J. R., and McLaughlin, S. W., "Generalized partial response targets for perpendicular recording," in *Proc. of the IEEE International Conference on Magnetics (Intermag) 2002*, pp. GP–03, April 28 – May 2, 2002.

[41] Lay, N. E. and Polydoros, A., "Per-survivor processing for channel acquisition, data detection and modulation classification," *Conference Record of the Twenty-Eighth Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1169–1173, October 31 – November 2, 1995.

[42] Leon-Garcia, A., *Probability and random processes for electrical engineering*. New york: Addison-Wesley Publisher Company, Inc., second ed., 1994.

[43] Lottici, V. and Luise, M., "Embedding carrier phase recovery into iterative decoding of turbo-coded kiner modulation," *IEEE Trans. Commun.*, vol. 52, pp. 661–669, April 2004.

[44] Lottici, V. and Luise, M., "Iterative carrier phase synchronization for coherent detection of turbo-coded modulation," in *Proc. European Wireless*, p. 140, February 2002.

[45] Mallory, P., "A maximum likelihood bit synchronizer," *International Telemetrin Conf., Proc., IV (1968)*, pp. 1–16, 1968.

[46] Mengali, U. and Andrea, A. N., *Synchronization techniques for digital receivers*. New York: Plenum Press, 1997.

[47] Messerschmitt, D. G., "Design of finite impulse response for the viterbi algorithm and decision-feedback equalizer," in *Proc. of ICC'74*, pp. 37D–1–5, June 1974.

[48] Meyr, H., Moeneclaey, M., and Fechtel, S. A., *Digital Communication Receivers: Synchronization, Channel Estimation, and Signal Processing*. New York: John Wiley & Sons, Inc., 1997.

[49] Mielczarek, B., "Synchronization in turbo coded systems," Tech. Rep. 342L, Department of Signals and Systems, Chalmers University of Technology, Göteborg, Sweden, April 2000.

[50] Moision, B. E., Siegel, P. H., and Soljanin, E., "Distance-enhancing for digital recording," *IEEE Trans. Magnetics*, vol. 34, pp. 68–74, January 1998.

[51] Moon, J. and Park, J., "Pattern-dependent noise prediction in signal-dependent noise," *IEEE J. Selected Areas Comm.*, vol. 19, pp. 730–743, June 2001.

[52] Moon, J. and Carley, L. R., "Performance comparison of detection methods in magnetic recording," *IEEE Trans. Magnetics*, vol. 26, pp. 3155–3172, November 1990.

[53] Moon, J. and Zeng, W., "Equalization for maximum likelihood detector," *IEEE Trans. Magnetics*, vol. 31, pp. 1083–1088, March 1995.

[54] Mueller, K. H. and Müller, M., "Timing recovery in digital synchronous data receivers," *IEEE Trans. Commun.*, vol. COM-24, pp. 516–531, May 1976.

[55] Nayak, A. R., *Iterative timing recovery for magnetic recording channels with low signal-to-noise ratio*. PhD thesis, Georgia Institute of Technology, Georgia, June 2004.

[56] Nayak, A. R., Barry, J. R., and McLaughlin, S. W., "Joint timing recovery and turbo equalization for coded partial response channels," *IEEE Trans. Magnetics*, vol. 38, pp. 2295–2297, September 2003.

[57] Noels, N., Herzet, C., Dejonghe, A., Lottici, V., Steendam, H., Moeneclaey, M., Luise, M., and Vandendorpe, L., "Turbo synchronization: An EM algorithm interpretation," in *Proc. of ICC'03*, vol. 4, pp. 2933–2937, May 2003.

[58] Nuriyev, R. and Anastasopoulos, A., "Analysis of joint iterative decoding and phase estimation for the noncoherent AWGN channel using density evolution," *IEEE International Symposium on Information Theory,* Lausanne, Switzerland, p. 168, June 30 – July 5, 2002.

[59] Oenning, T. R. and Moon, J., "Partial response maximum likelihood detection for perpendicular recording," *IEEE International Conference on Magnetics (INTERMAG) 2000*, pp. HT–08, 2000.

[60] Oh, W. and Cheun, K., "Joint decoding and carrier phase recovery algorithm for turbo codes," *IEEE Communications Letters*, vol. 5, pp. 375–377, September 2001.

[61] Otnes, R. and Tüchler, M., "Exit chart analysis applied to adaptive turbo equalization," in *Proc. Nordic Signal Processing Symposium*, October 2002.

[62] P. R. Chevillat, E. E. and Maiwald, D., "Noise-predictive partial-response equalizers and applications," in *Proc. of ICC'92*, vol. 2, pp. 942–947, June 1992.

[63] Qureshi, S. U. and Forney, G. D., "Performance and properties of a $T/2$ equalizer," *Conference Record, NTC 1977*, December 1977.

[64] Raghaven, S. and Thapar, H. K., "Feed-forward timing recovery for digital magnetic recording," in *Proc. of ICC'91*, vol. 2, pp. 794–798, June 1991.

[65] Raheli, R., Polydoros, A., and Tzou, C.-K., "The principle of per-survivor processing: a general approach to approximate and adaptive MLSE," in *Proc. of Globecom'91.*, vol. 2, pp. 1170–1175, December 1991.

[66] Raheli, R., Polydoros, A., and Tzou, C.-K., "Per-survivor processing: a general approach to MLSE in uncertain environments," *IEEE Trans. Commun.*, vol. 43, pp. 354–364, Feb/Mar/Apr 1995.

[67] Raphaeli, D. and Zarai, Y., "Combine turbo equalization and turbo decoding," in *Proc. of Globecom'97*, vol. 2, pp. 639–643, November 1997.

[68] Rizos, A. D. and Proakis, J. G., "Reduced-complexity sequence detection approaches for PR-shaped, coded linear modulations," in *Proc. of Globecom'97*, vol. 1, pp. 342–346, November 1997.

[69] Roscamp, T. A., Boerner, E. D., and Parker, G. J., "Three-dimensional modeling of perpendicular recording with soft underlayer," *J. of Applied Physics*, vol. 91, May 2002.

[70] Seshadri, N. and Anderson, J. B., "Decoding of severely filtered modulation codes using the (M,L) algorithm," *IEEE J. Selected Areas Commun.*, vol. 7, pp. 1006–1016, August 1989.

[71] Shafiee, H., "Timing recovery for sampling detectors in digital magnetic recording," in *Proc. of ICC'96*, vol. 1, pp. 577–581, January 1996.

[72] Simmons, J. B. and Mohan, S., "Sequential coding algorithms: A survey and cost analysis," *IEEE Trans. Commun.*, vol. COM-32, pp. 169–176, February 1984.

[73] Simmons, S. J., "Breadth-first trellis decoding with adaptive effort," *IEEE Trans. Commun.*, vol. COM-38, pp. 3–12, January 1990.

[74] Simmons, S. J. and Senyshyn, P., "Reduced-search trellis decoding of coded modulations over ISI channels," in *Proc. of Globecom'90*, vol. 1, pp. 393–396, December 1990.

[75] Souvignier, T., Friedmann, A., Öberg, M., Siegel, P. H., Swanson, R. E., and Wolf, J. K., "Turbo decoding for PR4: parallel vs. serial concatenation," in *Proc. of ICC'99*, vol. 3, pp. 1638–1642, June 1999.

[76] Steendam, H., Noels, N., and Moeneclaey, M., "Iterative carrier phase synchronization for low-density parity-check coded system," in *Proc. of ICC'03*, vol. 5, pp. 3120–3124, May 2003.

[77] Suzuki, T., "Perpendicular magnetic recording: Its basics and potential for the future," *IEEE Trans. Magnetics*, vol. MAG-20, pp. 675–680, September 1984.

[78] Tan, J. and Stüber, G. L., "New SISO decoding algorithms," *IEEE Trans. Commun.*, vol. 51, pp. 845–848, June 2003.

[79] Tüchler, M., Koetter, R., and Singer, A., "Turbo equalization: Principles and new results," *IEEE Trans. Commun.*, vol. 50, pp. 754–767, May 2002.

[80] ten Brink, S., "Convergence behavior of iteratively decoded parallel concatenated codes," *IEEE Trans. Commun.*, vol. 49, pp. 1727–1737, October 2001.

[81] Thapar, H. K. and Patel, A. M., "A class of partial response systems for increasing storage density in magnetic recording," *IEEE Trans. Magnetics*, vol. 23, pp. 3666–3668, September 1987.

[82] Ungerboeck, G., "Fractionally tap-spacing equalizer and consequences for clock recovery in data modem," *IEEE Trans. Commun.*, vol. COM-24, pp. 856–864, August 1976.

[83] Vanelli-Coralli, A., Salmi, P., Cioni, S., Corazza, G. E., and Polydoros, A., "A performance review of PSP for joint phase/frequency and data estimation in future broadband satellite networks," *IEEE J. Selected Areas Commun.*, vol. 19, pp. 2298–2309, December 2001.

[84] Walsh, J. W., Johnson, C. R., and Regalia, P. A., "Joint synchronization and decoding exploiting the turbo principle," in *Proc. of the 38th Conference on Information Sciences and Systems, 2004*, pp. 17–19, March 2004.

[85] Wang, S. X. and Taratorin, A. M., *Magnetic Information Storage Technology*. San Diego: Academic Press, 1999.

[86] Wicker, S. B., *Error control systems for digital communication and storage*. Upper Saddle River, New Jersey: Printice Hall International, 1995.

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

[88] Zhang, L. and Burr, A. G., "A new method of carrier phase recovery for BPSK system using turbo-codes over AWGN channel," in *Proc. of the 12th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications 2001*, vol. 1, pp. A179–A183, October 2001.

VITA

Piya Kovintavewat was born in Bangkok, Thailand, in 1972. He received the B.Eng. degree summa cum laude from Thammasat University, Thailand, in March 1994. After graduation, he worked as an engineer at Thai Telephone and Telecommunication company, one of the country's largest telephone operators, for three and a half years. In September 1997, he was granted a scholarship by the Swedish Foundation for International Cooperation in Research and Higher Education to study for the M.Sc. degree at Chalmers University of Technology, Göteborg, Sweden. After receiving the M.Sc. degree in November 1998, he joined the National Electronics and Computer Technology Center, a dynamic organization responsible for the development of Information Technology in Thailand. As a research assistant, he was involved in two main projects, namely, public-key infrastructure and Thailand smart card standard. In January 2000, he received a scholarship from the royal Thai government to pursue the Ph.D. degree in electrical engineering at Georgia Institute of Technology, Atlanta, GA, USA. During his Ph.D. stay, he was partly supported by Seagate Technology under the guidance of Prof. John R. Barry. He also obtained work experiences with Seagate Technology, Pittsburgh, PA, USA, in the summers of 2001, 2002, and 2004. He received the Ph.D. degree in December 2004. He will join Nakhon Pathom Rajabhat University, Thailand, as an Assistant Professor, whose responsibilities include educating students, conducting researches, participating in the planning process of educational plans, and developing the national science and technology infrastructure. His main research interests include coding and signal processing techniques for data storage systems and wireless communication systems, with emphasis on timing recovery, equalization, and error-correction decoding.