Iterative Timing Recovery for Magnetic Recording Channels with Low Signal-to-Noise Ratio

A Thesis
Presented to
The Academic Faculty

by

Aravind R. Nayak

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

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

Iterative Timing Recovery for Magnetic Recording Channels with Low Signal-to-Noise Ratio

Approved by:

Dr. John R. Barry, Advisor

Dr. David G. Taylor

Dr. Steven W. McLaughlin, Co-Advisor

Dr. Thomas D. Morley

Dr. Gordon L. Stüber

Date Approved: 24 June 2004

To Dilip, Antony and Clement.

ACKNOWLEDGEMENTS

I thank Dr. John R. Barry and Dr. Steven W. McLaughlin for being excellent advisors and great people. They have guided me through the last five years with skill and patience, and I am the better for it. Working with them has taught me the importance of attention to detail, of clarity of thought and expression, and of not mistaking the trees for the woods. Thanks also for the opportunity to spend a year in GT Lorraine, France! I also thank Dr. Gordon L. Stüber, Dr. David G. Taylor and Dr. Thomas D. Morley for serving on my defence committee. Thanks are also due to Zak Keirn and German Feyh for the invaluable opportunity to work with them in Summer 2003. The work was great, and so was the chance to be in Colorado!

Now, things start to get a bit murky, what with so many other people to thank! As with anyone else and to an even greater extent, I have been helped and supported along the way by a great number of people and it is impossible for me to thank everyone of them without the risk of having the acknowledgements section as the biggest portion of this document. So, the following is a partial list of people whom I thank in the most heartfelt way.

Thanks to Dilip, Antony and Clement for being who they are. I learnt an enormous lot by being with them, listening to them, and talking to them during my days at Madras and even after. Thanks also to the whole gang at Madras: Badri, Chava, Praveen, Kiran and Sabu.

Life would have been much the worse for me if it had not been for Niranjan and Ashwini. Thank you, Niranjan, for being so full of life, with a never a dull moment with you around. Thank you, Ashwini, for the roller-coaster rides, and also for opening up a whole new chapter in my life and more importantly agreeing to be

a part of it! And of course, thanks to mother and Ajanth for being so unvaweringly supportive with more confidence in me than safely warranted.

Thanks also to the staff at Georgia Tech, notably Cordai Farrar and Marilou Mycko, who were very helpful and took a lot of stress out of the administrative details of the PhD process, traveling to conferences, etc.

I can not thank the Georgia Tech gang enough. The five years I spent at Tech have truly been the most enjoyable I have had and they have played no mean part in it. Thanks to the Tumlin gang: Badri, Pradnya, Shilpa, Avanti, Srini and Nisarga for a whale of a time with all the fun-filled activities. Thanks also to the GCATT gang: Andrew, Renato (and Turia), Badri, Shayan, Joon-Hyun, Piya, Arumugam, Estuardo, Ravi, Rajesh and Shantanu. All the discussions about topics technical and non-technical that we had were fun and enriching. Special thanks to Renato and Turia for the great trip to the Rockies, and to Renato for patiently going through my thesis and the final presentation and giving detailed and helpful comments. Thanks to Andrew for his infinite wisdom, the great movies and his famous secret for good cooking. And not to mention, thanks to Shayan and Rajesh for putting up with me the homeless! By this point, the keen reader would undoubtedly have noticed the presence of Badri in almost all the lists I mentioned so far. Forces beyond our control have made sure that for the last nine years, both of us have had to endure each other's company, be it in Madras, Atlanta or Metz. Not too sure it won't happen again in the future, so I have decided to be nice to Badri and thank him specially! Kidding aside, I whole-heartedly thank Badri. Thanks also to Poonam, Suchi and Deepak Jahagirdar. Though I got to know them well relatively recently, I can honestly say that it has been a great pleasure. I could go on like this, but will stop with thanking all those who put up with my incessant requests for playing badminton and also actually obliged me!

TABLE OF CONTENTS

LIST OF FIGURES

SUMMARY

Digital communication systems invariably employ an underlying analog communication channel. At the transmitter, data is modulated to obtain an analog waveform which is input to the channel. At the receiver, the output of the channel needs to be mapped back into the discrete domain. To this effect, the continuous-time received waveform is sampled at instants chosen by the timing recovery block.

A widely used timing recovery method is based on a phase-locked loop (PLL), which updates its timing estimates based on a decision-directed device. Timing recovery performance is a strong function of the reliability of decisions, and hence, of the channel signal-to-noise ratio (SNR). Iteratively decodable error-control codes (ECCs) like turbo codes and LDPC codes allow operation at SNRs lower than ever before, thus exacerbating timing recovery.

We propose iterative timing recovery, where the timing recovery block, the equalizer and the ECC decoder exchange information, giving the timing recovery block access to decisions that are much more reliable than the instantaneous ones. This provides significant SNR gains at a marginal complexity penalty over a conventional turbo equalizer where the equalizer and the ECC decoder exchange information. We also derive the Cramér-Rao bound, which is a lower bound on the estimation error variance of any timing estimator, and propose timing recovery methods that outperform the conventional PLL and achieve the Cramér-Rao bound in some cases.

At low SNR, timing recovery suffers from cycle slips, where the receiver drops or adds one or more symbols, and consequently, almost always the ECC decoder fails to decode. Iterative timing recovery has the ability to corrects cycle slips. To reduce the number of iterations, we propose cycle slip detection and correction methods.

CHAPTER 1

INTRODUCTION

Timing recovery is an essential component of digital communication systems. Most physical communication channels are analog in nature. The information to be transmitted is in the form of bits, which are modulated to suit the analog channel characteristics. The output of the channel needs to be mapped back into the discrete domain at the receiver. To this end, the continuous-time received waveform is sampled at instants chosen by the timing recovery block. The system error-rate varies drastically with the performance of timing recovery, which, in turn, is a strong function of the operating SNR.

A widely used timing recovery method is the decision-directed phase-locked loop (PLL) [64]. The PLL-based timing recovery scheme employs a timing error detector (TED) to estimate the timing error, and uses these estimates to correct its timing estimates. The TED uses previous received samples and decisions on the previous transmitted symbols based on the previous samples to generate timing error estimates. The feedback loop thus formed is sensitive to the delay in the loop, and the decision latency can not be too large. With high SNR, instantaneous decisions are reliable enough for the timing recovery mechanism to perform well.

The introduction of iteratively decodable error control codes (ECCs) like turbo codes [9] and low-density parity check (LDPC) codes [25] led to a significant reduction in the operating SNR due to the large coding gains afforded by the iterative decoding of these codes. In addition, the principle of iterative decoding has been extended to include equalization as well. Turbo equalization [54], where the equalizer and the ECC decoder iterate, allows an even lower SNR of operation. Thus, the timing

recovery block now needs to operate at SNR lower than ever before.

The decisions from the ECC decoder are more reliable than the instantaneous ones, but usually the decoding process introduces significant latency. At high SNR, we can afford to use instantaneous decisions, thus avoiding decision latency altogether. At low SNR, however, we need to take the presence of the ECC into account in order to have reliable decisions during timing recovery.

The maximum-likelihood solution to this problem would require the joint timing recovery, equalization and decoding. This is prohibitively complex. The expectation-maximization algorithm [26] provides an iterative solution to this problem, but even this solution has very high complexity.

We propose iterative timing recovery as a low-complexity solution to the problem of timing recovery at low SNR. Iterative timing recovery is an extension of the turbo equalization principle to include the timing recovery block, facilitating exchange of information between the timing recovery block, the equalizer and the ECC decoder. At the end of each turbo equalizer iteration involving the equalizer and the ECC decoder, decisions more reliable than the instantaneous ones are available, and these can be fed back to the timing recovery block to improve its performance. These better timing estimates can then be used to refine the samples and this, in turn, improves the performance of the turbo equalizer in the next iteration. As the complexity of the timing recovery operation is usually negligible compared to that of a turbo equalizer iteration, the increase in complexity per iteration is minimal.

So far, to improve the low SNR performance of the system, we used the presence of the ECC. Without changing the timing recovery architecture, system performance was enhanced by using better decisions available from the ECC decoder. An alternative approach is to improve the timing recovery architecture itself. To this end, we first derive the Cramér-Rao bound (CRB) [63] for the timing recovery problem. This is a lower bound on the estimation error variance for any unbiased timing recovery

method. We compare the conventional PLL-based timing recovery method with the CRB, and observe that the PLL-based system does not achieve the CRB. Next, we propose algorithms that achieve the CRB in the presence of a frequency offset, and for the random walk case, we present an maximum *a posteriori* (MAP) [63] estimation-based algorithm that outperforms the PLL.

A major problem with low SNR and moderate to severe timing offset models is the phenomenon of cycle slips [64], where the receiver *adds* or *drops* a symbol entirely, leading to catastrophic errors. In the presence of cycle slips, the ECC decoder on its own almost always fails to decode. An advantage of iterative timing recovery is that it is capable of correcting cycle slips. When a cycle slip occurs, the timing recovery unit does not suddenly add or drop symbols. Rather, it loses track of the actual timing offsets gradually, eventually settling down at an offset corresponding to multiples of the symbol duration. As iterations progress, the size of the boundary zone between perfect lock and multiple symbol offset reduces and this boundary moves towards the end of the packet. Also, as iterations progress, the size of the portion of the packet affected by this offset reduces, and eventually, the cycle slip is *pinched out*. To reduce the number of iterations needed, we propose some simple cycle slip detection and correction methods. In addition, we show that splitting the acquisition preamble into two halves and placing these at the beginning and at the end of the packet greatly reduces the occurrence of cycle slips. When faced with a frequency offset timing model, this arrangement minimizes the CRB on the estimation of the frequency offset and also of the initial timing offset.

For multi-parameter estimation problems, usually the joint estimation problems is a hard one. Iterative solutions that involve exchange of information between sub-optimal estimators, each of which estimates a subset of the parameters, as opposed to all the parameters jointly, approximate the joint estimation solution very well in many cases. This has led to the conjecture that it is the process of iteration itself

that provides most of the power of these solutions, as opposed to the optimality of the component estimator blocks. In other words, in an iterative setting, it is not critical to optimize the individual blocks. Given reasonably good individual blocks, the process of iteration gets us close to the best possible performance. The research presented in this thesis support this conjecture. When we use the error-rate as the metric of comparison, using the PLL as the timing recovery block as opposed to using the CRB-achieving methods leads to very little degradation in performance. Much of the loss of the PLL-based iterative system with respect to the best possible performance is due to the occurence of cycle slips. The performance of a receiver with known timing offsets provides a lower bound to the performance of timing recovery methods. In addition, another lower bound is provided by the genie-aided receiver, whose timing recovery block has access to the transmitted data. Used in conjunction with the preamble placement to minimize the CRB, the PLL-based iterative timing recovery method performs close to the genie-aided system with reasonable complexity.

The structure of the rest of the thesis is as follows. In Chapter 2, we discuss the magnetic recording channel that forms the basis of all the examples and simulation results. In addition, we set up the timing recovery problem and review the PLL-based timing recovery method. We detail the conventional coded system that uses iterative ECCs *i.e.*, turbo codes and LDPC codes. We also discuss turbo equalization, based on which we propose iterative timing recovery. In addition, we present previous work in literature regarding iterative synchronization. In Chapter 3, we derive the CRB for various timing models, namely constant offset, frequency offset, random walk and a combination of these. In Chapter 4, we discuss the conventional PLL-based timing recovery method for magnetic recording channels and compare it to the CRB. In Chapter 5, we propose the gradient search based methods to achieve the CRB in the case of the frequency offset, and also the MAP algorithm for the random walk case. In Chapter 6, we discuss the problem of acquisition. We prove that to minimize the CRB,

we need to keep half the known symbols at the beginning of the packet and half at the end. Also, we demonstrate by simulation that this arrangement greatly reduces the occurrence of cycle slips when we have a frequency offset timing model. In Chapter 7, we propose iterative timing recovery, and also present simulation results to show the gains that can be achieved using iterative timing recovery and the associated cycle slip detection and correction methods. Finally, we present conclusions and future work in Chapter 8.

CHAPTER 2

BACKGROUND INFORMATION

In this chapter, we present background information on the magnetic recording channel, the timing recovery problem, the conventional coded system considered in this thesis, and also on iterative approaches to synchronization in the literature.

In this thesis, we consider the general problem of timing recovery for channels with inter-symbol interference (ISI) and baud-spaced sampling. The magnetic recording system that we consider is an example of such a channel, and a significant portion of the simulation results presented here are based on the magnetic recording channel. The bounds and the algorithms presented in the following chapters, however, are more general and are not limited to the magnetic recording channel.

In Section 2.1, we give a brief overview of digital magnetic data storage based on hard drives. In Section 2.2, we give a brief overview of the timing recovery problem and review the conventional timing recovery method used in hard drives. In Section 2.3, we present a description of the conventional coded system that uses iterative error-control codes, and also of turbo equalization. In Section 2.4, a summary of related work on iterative synchronization is provided. Finally, the chapter is summarized in Section 2.5.

2.1 Magnetic Recording

An important factor in the explosion of computational abilities that is being witnessed today is the ability to store and access large amounts of data quickly and efficiently. Modern computers and, increasingly, a large number of other devices as well, use digital magnetic storage for this purpose. Digital magnetic storage involves storing

data based on the direction of magnetization of magnetic media. In the following subsections, we present a brief overview of magnetic recording based on material in [8], [32], and [15]. The reader is referred to the following references for more detailed expositions: [8], [32], [38], [28].

2.1.1 History

Earliest recording techniques for computers involved paper, in the form of either punch cards or paper tape, where the information was stored by either punching or writing on paper [32]. This method was painstaking and slow, and the medium was not very durable. The next big advance was the introduction of magnetic tapes, where data was recorded and read out in a fashion similar to that of an audio tape. The main disadvantage with magnetic tapes is that the data can be accessed only in a linear fashion, as opposed to the more desirable random access. Floppy disks, introduced next, allowed random access, though the storage capacity and access speed were still relatively low.

The earliest experimental disk drives were rotating cylindrical drums, coated with magnetic material, on which the data was stored [32]. An improvement over this was the hard disk drive, where the data was stored on a magnetic disk, and data was written and read out using an electro-magnetic head. A main disadvantage with this technology was that the head was in contact with the medium leading to wear and tear. The innovation that literally allowed magnetic storage technology to take off was the realization that it is possible to suspend the head on the medium with an air gap and still be able to read the data stored in the medium as the head flies over the rotating medium. The first production hard drive based on this principle was introduced in 1956 by IBM and was called the IBM 305 RAMAC[32].

Further advances led to reduction in the air gap size, better magnetic heads, more aerial density for the information bits, and all of these factors led to a significant

reduction in the size of the drives, and at the same time, a great increase in the data storage capacity of the drives. Currently, hard drives are the dominant medium of data storage, with almost every PC equipped with one. Of late, hard drives have started to appear in hand-held devices and consumer electronic devices as well.

2.1.2 Operational Overview

Hard drives store information on circular magnetic disks, also known as platters. Each platter has magnetic material coated on both the surfaces. On each surface, data is stored along concentric circles called tracks. Each track is further subdivided into sectors, each of which holds 512 bytes of data.

The platters are mounted by cutting a hole through their centers and stacking them up on a spindle. The spindle is attached to a spindle motor which rotates the platters at a high speed. Data is written to and read from the medium by an electromagnetic device called the head. Multiple heads are employed, with each surface being accessed by a head devoted to it. The head flies over the medium as it rotates, and reacts to the magnetization of the medium. During the write process, information to be stored is transformed into an electrical signal and fed to the head, which in turn magnetizes the underlying medium. During the read process, the head interacts with the magnetic field of the medium and produces an electrical signal which can then be converted back into the data bits.

Earlier hard drives, called ferrite heads, used a coil to convert the signals between the electrical and the magnetic domains. Ferrite heads are well-suited for the write process because the strength of the magnetic field can be increased by simply increasing the number of coils. But doing this, conversely, makes the read process difficult. Newer drives use magneto-resistive (MR) heads for the read process, and this allows much higher areal data storage densities. MR heads use certain materials that change their resistance based on the magnetic field they are subjected to are used. MR heads

are much more sensitive to changes in magnetic fields and are thus better suited to the read process. Therefore, the read and the write processes are assigned to two different heads: an MR head for reading, and a ferrite head for writing.

Based on the giant magneto-resistive (GMR) property, a new class of heads known as GMR heads was introduced in 1997, and this led to a further increase in the areal storage density [32]. In summary, magnetic storage technology has been steadily improving, providing better storage devices, and entering newer application markets.

From a communication point of view, the read and the write processes introduce ISI, and thus can be seen as a channel. We need to perform equalization at the receiver to combat inter-symbol interference (ISI) due to the channel. In addition, we have run-length limited (RLL) encoding, error-control coding (ECC), and precoding at the transmitter. And at the receiver, we need to undo the effect of all the three operations to finally arrive at the user data.

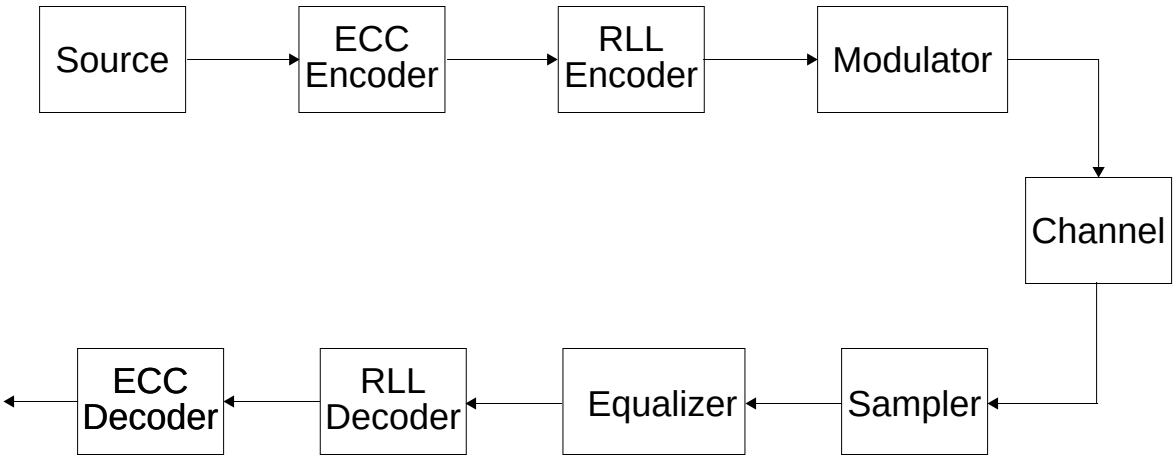

Figure 1: System block diagram from a communications point of view.

The block diagram of a magnetic data storage system from the communications point of view is shown in Figure 1. The data from the source or the user are first encoded by the ECC encoder. The ECC encoded bits are then encoded by an RLL encoder to satisfy the run-length constraints of the system. These RLL-encoded bits

are then precoded if necessary. Finally, the discrete-time information in the form of the precoded bits is converted into a continuous-time signal suitable to be put on to the channel. At the receiver, we perform sampling and equalization to undo the channel distortion and get back from the continuous-time domain to the discrete-time domain. For simplicity, in the block diagram, we have shown the sampling preceding the equalization. In practice, part of the equalization is done before the sampling, as explained in the following subsections. After equalization, RLL decoding and ECC decoding are performed to finally arrive at the receiver's estimate of the transmitted user information.

In the following subsections, we look into these components in slightly more detail.

2.1.3 Write

Although Magnetic recording is inherently a non-linear process, it can be made approximately linear by clever signal processing [8]. Consider, for instance, longitudinal recording, where a signal current is applied to the magnetic head and the resulting flux magnetizes the medium longitudinally, i.e., along or against the direction of relative motion of the head and the medium. If the input signal is a binary signal of sufficient amplitude, the medium is magnetized to saturation in one of two opposite directions. In this case, the write process is essentially linear. For this reason, practical magnetic storage systems use binary saturation recording, where the medium is magnetized to saturation along one of two directions.

The write process is fraught with non-idealities like non-zero transition width, hysteresis, demagnetizing fields due to adjacent transitions, overwrite noise and transition noise [8]. As a result, the transitions, where the magnetization changes direction, are randomly shifted with respect to the ideal locations, and also are zig-zag instead of being straight lines. When adjacent zig-zag transitions come into contact, partial erasures result. Random shifts of the transitions degrade the signal-to-noise ratio

(SNR) of the channel.

2.1.4 Read-back

The read-back process with binary saturation recording is also essentially linear [8]. During read-back, the head moves over the medium and responds to the transitions in the direction of magnetization. For an isolated transition, it produces a pulse $g(t)$ or its inverse $-g(t)$ depending on the direction of magnetization. The pulse $g(t)$ is called the transition response and is usually modeled as a Lorentzian pulse given by

$$g(t) \quad = \quad \frac{T}{\pi t_{50}} \frac{1}{1 + \left(\frac{2t}{t_{50}}\right)^2},$$

where t_{50} is the pulse width of $g(t)$ at half amplitude, and T is the symbol duration, i.e., the minimum spacing between two transitions. The parameter $D = t_{50}/T$ is the information density and is a normalized measure of how many bits are packed into the *resolution unit* t_{50}. The response of the head to a single isolated symbol, consisting of an two transitions spaced by T, is $f(t) = g(t) - g(t - T)$ and is called the symbol response.

An important parameter that characterizes a magnetic medium is the minimum mark size, which is the minimum separations required between successive transitions for these to be distinguishable for the detector. This parameter depends on both the write and the read mechanisms employed.

2.1.5 Equalization

The traditional method of reading data from a hard drive is the peak detect method. The detector detects magnetic flux reversals in the channel, and this results in voltage spikes in the electrical circuit. This method works as long as the spikes are far apart and strong enough when compared to the background noise. As the data densities increase, these conditions are no longer satisfied, leading to SNR degradation and ISI.

Magnetic channels are band-pass, exhibiting a null at dc. Therefore, full-response equalization leads to noise enhancement at low frequencies. The partial response maximum likelihood (PRML) technique gets around this problem by breaking the equalization down into two steps [15]. First, we equalize to a target that allows controlled ISI. An example of a controlled-ISI channel is the class-IV partial response (PR) target, also called the PR-IV response. The impulse response of the PR-IV channel is represented by the polynomial $1 - D^2$, where D is the delay operator. In words, the PR-IV symbols d_k are given by $d_k = a_k - a_{k-2}$, where $\{a_k\}$ is the data written into the channel.

The read-back waveform is filtered by a front-end filter to eliminate the out-of-band noise, and equalized to a partial response target to result in

$$r(t) = \sum a_k h(t - kT - \tau_k) + n(t), \tag{1}$$

where T is the symbol duration, $a_k \in \{-1, +1\}$ are the precoded symbols (precoded to combat error propagation), $h(t)$ is the pulse shape corresponding to the controlled-ISI target, $n(t)$ is Gaussian noise band-limited to $[-1/2T, 1/2T]$, and τ_k is the unknown timing offset for the k^{th} symbol. This completes the first stage of PRML equalization. The waveform $r(t)$ is then sampled by the timing recovery block to produce samples $\{r_k\}$.

Then, maximum likelihood sequence detection (MLSD) using the Viterbi algorithm [23] is implemented on the trellis resulting from the controlled ISI. Noise enhancement is greatly reduced since the partial response target exhibits a null at dc and at the Nyquist frequency, and also in the frequency domain, its shape is similar to the channel response itself. Furthermore, for the PR-IV target, MLSD achieves the matched-filter bound and therefore, the resulting error probability is similar to that with no ISI [8].

2.1.6 Timing Recovery

After the first step of equalization to a PR target, the waveform $r(t)$ is then sampled at time instants $kT + \hat{\tau}_k$ chosen by a timing recovery device, leading to samples r_k. These samples are then used to perform the ML equalization step to complete the equalization process. Details regarding the conventional timing recovery method used for magnetic recording are provided in Section 2.2.

2.1.7 Precoding

A portion of the equalization task can be shifted to the transmitter by precoding the data symbols at the transmitter. Instead of the data symbols themselves being written to the channel, we precode the data symbols and these new precoded symbols are written to the channel. With knowledge of the channel characteristics, the precoder can be chosen to partly undo the channel distortion, thus reducing the equalization burden at the receiver. Precoders can also be chosen to prevent catastrophic error propagation. For the PR-IV system, for example, in our simulations, we employ the precoder denoted by the polynomial fraction $1/(1 \oplus D^2)$ [24] [15]. The precoded symbols $\{b_k\}$ are arrived at from the data symbols $\{a_k\}$ according to $b_k = b_{k-2} \oplus a_k$.

2.1.8 Run-length Limited Codes

RLL codes are modulation codes, used to modify the bit pattern to suit the channel requirements. The earlier modulation scheme used were the frequency modulation (FM), or the modified frequency modulation (MFM) schemes [8]. With FM, the number of flux reversals for a '1' and a '0' are different. MFM is similar to FM in that different patterns are assigned different number of flux reversals, but it is more efficient. Even more efficient is the family of RLL codes, which forms the standard for today's hard drives.

Optical and magnetic recording channels are examples of constrained channels where the sequences written into the channel need to satisfy some constraints. A

common set of constraints is the family of (d, k)-constraints [37]. Sequences satisfying the (d, k)-constraint have at least d zeros and at most k zeros separating successive ones occurring in the sequence. These constraints reduce the ISI in the system by separating consecutive transitions, and also improve the timing recovery performance by assuring sufficient transitions.

Another advantage with using RLL codes is that they allow for a higher density of user bits. With a channel constrained to two levels, the additional restriction of uniform symbol durations further reduces capacity. In other words, with a two-level channel, pulse-width modulation has higher capacity when compared to a modulation scheme that uses uniform-width pulses. Pulse-width modulation can be approximately implemented using an RLL-encoded stream at an artificially high symbol rate. For example, with a $(3, 7)$-RLL code, the possible pulse-widths are $4, 5, 6, 7, 8$ symbols long, and the corresponding patterns are $1000, 10000, 100000, 1000000, 10000000$ respectively. Assigning the minimum mark size to the pattern 1000, we can achieve pulse-widths of $5L/4$, $3L/2$, $7L/4$ and $2L$, where L is the minimum mark size. Increasing d and k suitably, we can approximate the pulse-width modulation strategy better.

The state-splitting algorithm [2] provides a systematic way of constructing run-length limited codes with state-dependent encoders and sliding-block decoders. An alternative method of run-length limited encoding and decoding is using the enumerative approach for permutation codes [16].

2.1.9 Error Control Codes

Another type of encoding used is error control coding, used to mitigate the effects of noise introduced by the channel. Reed-Solomon (RS) codes are a common family of ECCs used in the recording industry [32]. RS codes are powerful algebraic codes defined by generator polynomials whose zeros are consecutive powers of a primitive

root in a finite field [67]. RS codes are good at handling erasures, and therefore, are useful in the recording industry where channel imperfections due to scratches can be effectively modeled as erasures. To further improve the performance with erasures, interleaved RS codes are used to distribute the erasures over multiple RS-encoded blocks.

Recently, iteratively decodable codes like turbo codes [9] and low-density parity-check (LDPC) [25] codes have been proposed for recording applications since their high coding gains allow for higher recording densities. We employ these codes in the coded system considered in this thesis. Details on these codes are provided in Section 2.3.

2.2 Timing Recovery

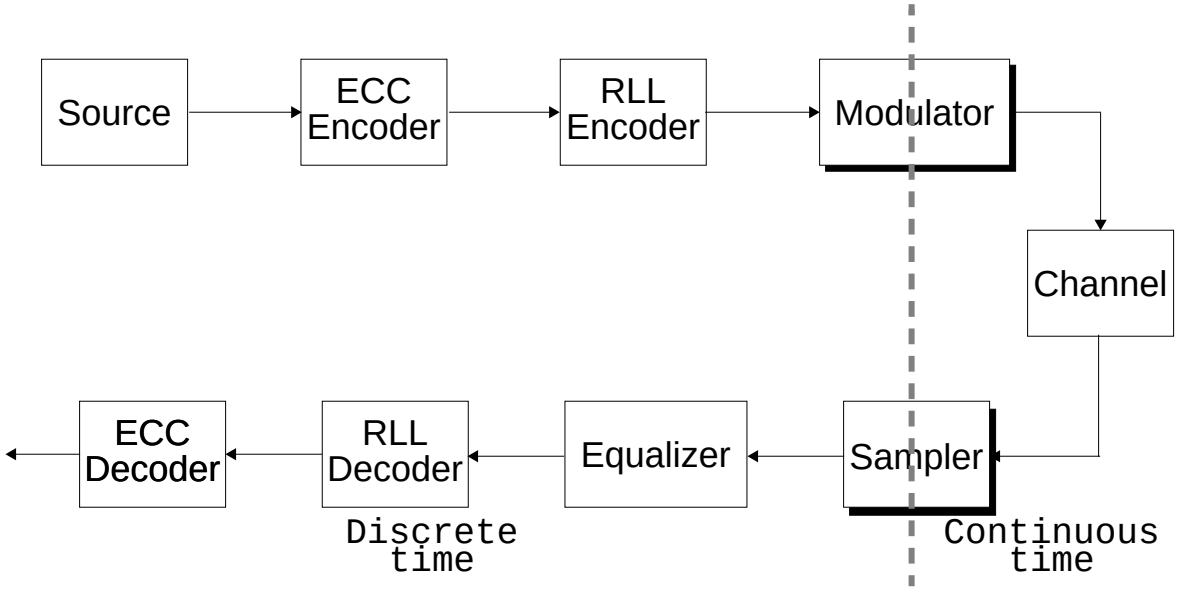

Figure 2: Continuous-time discrete-time interface.

Timing recovery is an essential part of digital communication and storage systems. The underlying physical channel is analog in nature whereas the user data is digital. At the transmitter, we obtain a continuous-time signal from the discrete-time user

data by modulation. From the sampling theorem, we know that it is sufficient to collect samples of this waveform taken so that the Nyquist criterion is satisfied. In other words, through appropriate sampling, we can undo the modulation performed and get back to the discrete-time domain. The interface between the discrete and the continuous domains is shown in Figure 2.

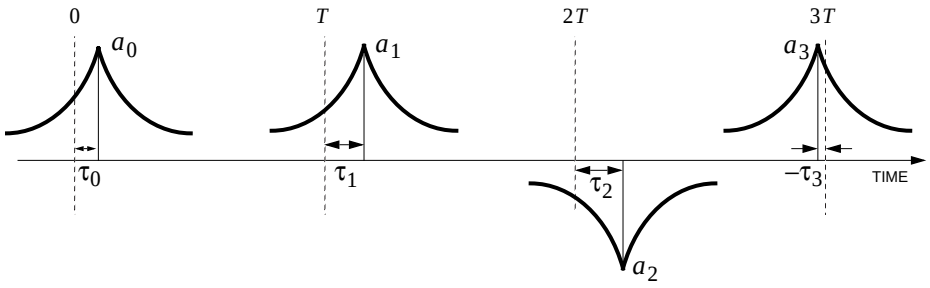

Figure 3: Timing jitter in communication systems.

Figure 3 shows an example of a modulation scheme used in communication systems. The modulator uses the data symbols $\{a_k\}$ to modulate a pulse shape $h(t)$, and the final waveform on the channel is obtained by shifting consecutive scaled pulses by the symbol duration T and adding them. We assume that $h(t)$ is band-limited such that baud-rate samples are sufficient statistics at the receiver. This modulation scheme is called pulse-amplitude modulation (PAM). Other techniques like frequency modulation exist, but we concentrate on PAM. Ideally, the pulses are all shifted with respect to each other by multiples of T, and therefore, at the receiver, it is sufficient to sample at multiples of T once we have the correct initial sampling instant. However, in practice, different pulses in the waveform suffer from timing jitter, with the k^{th} pulse having a timing offset of τ_k from the ideal location kT. The ideal sampling instants now are $kT + \tau_k$ instead of kT, and the timing recovery problem now is that of estimating the timing offsets $\{\tau_k\}$.

The maximum-likelihood (ML) timing estimator picks the timing offset estimates

$\hat{\tau}_k$ to minimize the cost function $J(\hat{\tau}; \boldsymbol{a})$, where

$$J(\hat{\boldsymbol{\tau}}; \boldsymbol{a}) = \int_{-\infty}^{\infty} (r(t) - \sum_{l} a_l h(t - lT - \hat{\tau}_l))^2 \, \mathrm{d}t, \tag{2}$$

where $r(t)$ is the received waveform. This represents the trained ML estimator, where the data symbols $\{a_k\}$ are assumed to be known at the receiver. The ML timing estimator is prohibitively complex except in a few simple cases. In general, suboptimal timing recovery methods are used that allow practical implementation at the cost of performance.

2.2.1 Classification

Timing recovery methods can be broadly classified in two different ways [39]. The first classification is in terms of structure: feedforward *vs.* feedback. Another classification is in terms of decisions: decision-aided *vs.* non-decision-aided. Combining these two modes of classification, we have four types of systems, and each of these types has its own set of advantages and disadvantages.

An important sub-category of timing recovery methods can be described as follows. The incoming signal is first passed through a non-linearity and the resulting output is either filtered using a narrow-band filter or fed to a phase-locked loop (PLL). With a narrow-band filter, we get the feedforward structure, and with the PLL, we have the feedback structure.

If the non-linear operation uses decisions about the transmitted data symbols, we have the decision-aided case, else the non-decision-aided case. An example of a non-decision-aided non-linearity is the squarer. For certain systems, a simple squaring of the incoming waveform generates a spectral line at the channel symbol rate, and this can then be extracted using a suitable narrow-band band-pass filter. Decision-aided non-linearities combine samples from the received signal with decisions about the transmitted symbols to generate estimates of the timing error, which can then be used in a PLL.

Another important classification of timing recovery methods is based not on the structure, but on the purpose they serve: acquisition *vs.* tracking. These represent the two phases in conventional timing recovery. At the beginning, the receiver has no information about the start of transmission. Also, the receiver has information about the nominal clock rate, and not the actual clock rate. During acquisition, the main tasks are to estimate the start of transmission, and the actual clock rate used at the transmitter. During tracking, these estimates are further refined and changes in the timing parameters are tracked.

Another classification is baud-spaced vs. fractionally-spaced, depending on the number of samples per symbol duration. In applications where the symbol rate is low enough, we can oversample the received signal to get multiple samples per symbol, and this allows for timing recovery methods that have low-complexity and good performance. In applications with high symbol rate, for example magnetic recording, the signals are oversampled to allow for roll-over of the channel impulse response. Oversampling further to aid timing recovery by having many samples per symbol is not feasible as this would lead to a reduction in the data storage density. In this case, we use baud-rate timing recovery methods.

For many applications, including magnetic recording, the decision-aided PLL-based method is the timing recovery method of choice in the tracking mode. For acquisition, either correlation methods, or trained (data-aided) PLL-based methods are used. Therefore, we focus our attention on the decision-directed PLL in the sequel. Keeping the magnetic recording channel in mind, we consider baud-spaced, decision-aided, PLL-based timing recovery as the conventional timing recovery method.

2.2.2 PLL-based Timing Recovery

In a conventional system, the different blocks shown in the block diagram of Figure 2 are designed independently, assuming that the other blocks function perfectly. For

timing recovery methods, the implication of this is the assumption of *i.i.d.* data symbols. With coding, however, the data symbols are not independent. However, for capacity-achieving codes being proposed for the magnetic recording systems, the *i.i.d.* assumption is approximately valid. Therefore, to a first order approximation, we can neglect coding, especially at high SNR, when there are few errors.

The block diagram of the uncoded system under consideration is shown in Figure 4. The channel output waveform $y(t)$ is given by

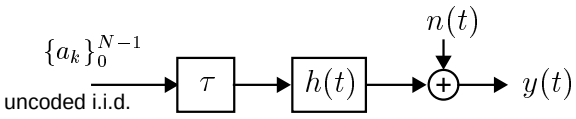

Figure 4: System block diagram with timing offsets, channel distortion and additive noise.

$$y(t) = \sum_{l=0}^{N-1} a_l h(t - lT - \tau_l) + n(t), \tag{3}$$

where T is the bit period, $a_l \in \{\pm 1\}$ are the N *i.i.d.* data symbols, $h(t)$ is the channel impulse response, $n(t)$ is additive white Gaussian noise, and τ_l is the unknown timing offset for the l^{th} symbol. After low-pass filtering $y(t)$ at the receiver to remove the out-of-band noise, the resulting continuous-time waveform $r(t)$ can be modeled as

$$r(t) = \sum_{l} a_l h(t - lT - \tau_l) + n_1(t), \tag{4}$$

where $n_1(t)$ is band-limited to $[-\frac{1}{2T}, \frac{1}{2T})$. The continuous-time waveform $r(t)$ is then sampled at timing instants $kT + \hat{\tau}_k$ based on the estimates $\{\hat{\tau}_k\}$ of $\{\tau_k\}$ produced by the timing recovery system. Ideally we would like to sample at instants $\{kT + \tau_k\}$.

Conventional timing recovery based on a PLL is shown in Figure 5. A first-order PLL updates its estimate of τ_k according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha \hat{\epsilon}_k, \tag{5}$$

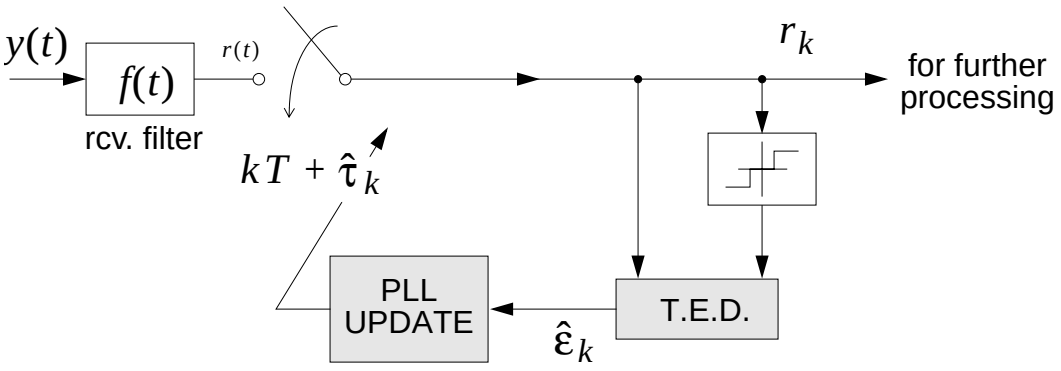

Figure 5: Conventional timing recovery.

where α is the PLL gain, and where $\hat{\epsilon}_k$ is the receiver's estimate of the estimation error $\epsilon_k = \tau_k - \hat{\tau}_k$. Timing error estimation is discussed in Section 2.2.3. The PLL update equation can be intuitively explained as follows. Assume that the timing offset is a constant. At the k^{th} sampling instant, the timing recovery block has access to the current estimate $\hat{\tau}_k$ and also an estimate of the timing error ϵ_k given by $\hat{\epsilon}_k$. If this estimate were accurate, then the receiver would simply add $\hat{\epsilon}_k$ to $\hat{\tau}_k$ to get τ_k and use this as $\hat{\tau}_{k+1}$ since this is the actual timing offset. However, in practice, what is available is a noisy estimate of ϵ_k, $\hat{\epsilon}_k$, and the PLL multiplies $\hat{\epsilon}_k$ by a factor α that determines the extent to which we attenuate the noise in the timing error estimate. A smaller α attenuates the noise better but leads to a larger rise-time while tracking a constant offset. Also, when tracking a time-varying timing offset, a smaller α limits the PLL's ability to track the variations. Therefore, in practice, α is chosen to strike a balance between these competing criteria: smaller rise time and good noise attenuation. A first-order PLL tracks a constant timing offset with zero steady-state error, but when faced with a frequency offset, it shows a non-zero steady-state error [64].

For zero steady-state error while tracking a frequency offset, we employ a second-order PLL which updates the estimate of τ_k according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha\hat{\epsilon}_k + \beta \sum_{l=0}^{k-1} \hat{\epsilon}_l, \tag{6}$$

where we have an additional gain parameter β. The accumulator output $\hat{\delta}_k = \sum_{l=0}^{k-1} \hat{\epsilon}_l$ tracks the average behavior of $\hat{\epsilon}_k$. A positive $\hat{\delta}_k$ implies that, on an average, $\hat{\epsilon}_l$ tends to be positive. In this case, it would be useful to add a positive quantity to the timing estimate $\hat{\tau}_{k+1}$ from a first order PLL. As $\hat{\epsilon}_k$ is a noisy version of ϵ_k, the gain parameter β helps attenuate the timing error estimation noise.

A second-order PLL gives us the advantage of being able to track a frequency offset with zero steady-state error, but it has the problem of possible instability. To illustrate this, consider the linearized model where the timing error detector (TED) that produces the timing error estimates $\hat{\epsilon}_k$ is assumed to be linear, *i.e.*, we assume that

$$\hat{\epsilon}_k = \epsilon_k + \nu_k, \tag{7}$$

where the noise terms $\{\nu_k\}$ are assumed to be *i.i.d.* and independent of $\{\epsilon_k\}$. Taking the z-transform of (6), the transfer function of the second-order PLL is given by

$$\hat{\tau}(z) = (\tau(z) + \nu(z)) \left[\frac{\alpha z + (\beta - \alpha)}{z^2 - (2 - \alpha)z + (1 - \alpha + \beta)} \right]. \tag{8}$$

The poles are of this transfer function are at $z = 1 - \frac{\alpha}{2} \pm \sqrt{\frac{\alpha^2}{4} - \beta}$. If β lies on the range $0 < \beta < \alpha$, these poles lie inside the unit circle and the timing loop is stable. With $0 < \beta < \frac{\alpha^2}{4}$, the system is under-damped and shows a slow rise-time. When $\beta > \frac{\alpha^2}{4}$, the system is over-damped and shows oscillatory behavior. The choice $\beta = \frac{\alpha^2}{4}$ represents the critically damped system that has both its poles at $1 - \frac{\alpha}{2}$. The fact that the linearized analysis guarantees a stable system if we ensure $0 < \beta < \alpha$ is not a very useful result as, in practice, the TED is not linear and the second-order PLL could become unstable even with $0 < \beta < \alpha$. For example, a second-order PLL, with

low enough SNR, occasionally diverges while tracking a frequency offset even with the gains being in the range suggested by the linearized analysis.

2.2.3 Timing Error Detector

Consider first the trained case when the transmitted symbols $\{a_l\}$ are available at the receiver. A timing error detector (TED) produces an estimate $\hat{\epsilon}_k$ based on channel observations $\{r_l\}$ and the data symbols $\{a_l\}$ [44] [15] [39]. Data symbols $\{a_l\}$ are used to generate noiseless, perfectly-timed samples $\{d_l\}$ and the TED generates $\hat{\epsilon}_k$ as a function of these samples $\{d_l\}$ and the actual samples $\{r_l\}$. The samples $\{d_k\}$ are given by

$$d_k = \sum_l a_l h(kT - lT). \tag{9}$$

In the absence of training, the TED operates in the decision-directed mode, where it uses decisions $\{\hat{a}_l\}$. Without training, we generate an estimate of d_k as

$$\hat{d}_k = \sum_l \hat{a}_l h(kT - lT). \tag{10}$$

Therefore, instead of symbols $\{a_k\}$ modulating the pulse shape $h(t)$, the channel can alternatively be viewed as symbols $\{d_k\}$ modulating the pulse shape $\mathrm{sinc}(t/T)$, which is a Nyquist pulse. For example, for the PR-IV channel, if $\{a_k\}$ are *i.i.d.* and equally likely to be $+1$ or -1, d_k takes on values -2, 0 and $+2$ with probabilities 0.25, 0.5 and 0.25 respectively. In the decision-directed case, one way to obtain the estimates $\{\hat{d}_k\}$ is by simply hard-quantizing the received samples $\{r_k\}$ to three values $\{-2, 0, +2\}$.

In the decision-directed case, the TED produces an estimate $\hat{\epsilon}_k$ based on channel observations $\{r_l\}$ and the estimates $\{\hat{d}_l\}$. In general, the TED equation can be written as

$$\hat{\epsilon}_k = f_k(\{r_l\}, \{\hat{d}_l\}), \tag{11}$$

where f_k is some function. The widely used Müller and Mueller (MM) TED [44] generates $\hat{\epsilon}_k$ according to

$$\hat{\epsilon}_k = r_k \hat{d}_{k-1} - r_{k-1} \hat{d}_k. \tag{12}$$

The MM TED can be made unbiased around the origin ($E[\hat{\epsilon}|\epsilon] = \epsilon$ for small ϵ) by introducing a scaling constant. For the PR-IV channel, assuming perfect decisions, the unbiased MM TED equation is

$$\hat{\epsilon}_k = \frac{3T}{16}\left(r_k\hat{d}_{k-1} - r_{k-1}\hat{d}_k\right). \tag{13}$$

The performance of TEDs can be evaluated using the so-called S-curve [44] [15], which is a plot of the average timing error estimate ($E[\hat{\epsilon}|\epsilon]$) as a function of the actual timing error (ϵ). Ideally, we would like this to be a straight line of unit slope passing through the origin. In addition to the mean of the error estimate (also called the timing function), we also plot the second moment of the error estimate. For the unbiased MM TED with the PR-IV channel and with ideal decisions, the timing function evaluates to [15]

$$\begin{aligned} E[\hat{\epsilon}|\epsilon] &= \frac{3T}{16}\left(h(-T+\epsilon) - 2h(T+\epsilon) + h(3T+\epsilon)\right), \\ &= \frac{3T}{16}\left(-p(-3T+\epsilon) + 3p(-T+\epsilon) - 3p(T+\epsilon) + p(3T+\epsilon)\right), \end{aligned} \tag{14}$$

where $p(t) = \mathrm{sinc}(t/T)$.

The timing function in the decision-directed case is arrived at by simulation. The normalized timing function (*i.e.*, $E[\hat{\epsilon}|\epsilon]/T$ *vs.* ϵ/T) and the second moment of the timing error for both these cases (labeled "Ideal" and "Hard" corresponding to the ideal and the hard-quantized decisions respectively) are plotted in Figures 6 and 7 respectively for SNR = 10dB. In the ideal case, the timing function is independent of the channel SNR and only the error variance depends on the SNR. In the decision-directed case, both the timing function and the error variance depend on the SNR [15]. The curves labeled "Soft" are for the case of soft decisions in the TED and will be discussed later in this section.

The ideal MM TED has a large linear range where the timing function can be approximated by a straight line of unit slope passing through the origin. The actual

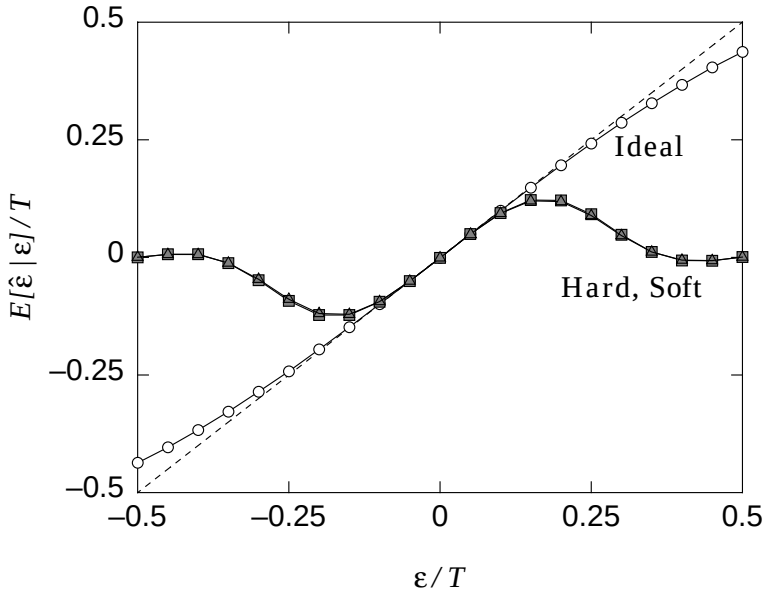

Figure 6: Timing function for the MM TED on the PR-IV channel.

timing error in the system is ϵ and $\hat{\epsilon}$ is the amount by which we correct the timing estimate. The decision-directed case shows linear behavior in the approximate range $\epsilon/T \in (-0.2, 0.2)$ beyond which $|\epsilon - \hat{\epsilon}|$ increases dramatically. Therefore, in the linear region, the TED (and the PLL) tracks the timing offsets well. Outside this region, as ϵ increases, $\hat{\epsilon}$ *decreases* and therefore, the PLL pulls farther and farther away from the actual timing offset. Zero crossings of the timing functions represent stable points of operation for the PLL and the decision directed MM TED has a zero crossing at around $\epsilon/T = 0.3$ and at $\epsilon/T = 0.5$ for the SNR considered above. Therefore, if we start with an offset of $\epsilon/T = 0.3$ at this SNR, for example, it is only noise perturbation and pattern-dependent jitter that can push us out of the equilibrium.

In addition to $\epsilon/T = 0.3$ and $\epsilon/T = 0.5$, the PLL also exhibits stable points of operation at $\epsilon = \pm T, \pm 2T, ...$ A transition from the origin to any of these, or a transition among these stable points, represents a *cycle slip*. An example of a cycle slip is shown in Figure 8, where $\hat{\tau}$ equals τ at the start of the packet, but by the end of the packet, $\hat{\tau}$ differs from τ by approximately $-T$. When a cycle slip occurs,

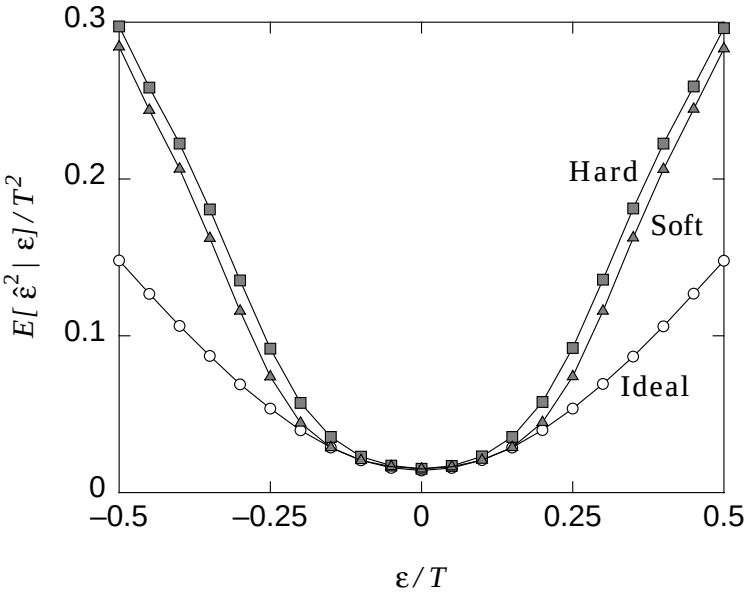

Figure 7: Timing error second moment for the MM TED on the PR-IV channel.

the receiver gains or loses data symbols and this has disastrous consequences for the processing that follows. The PLL operates in a bimodal fashion where it is in the linear mode around a stable operating point for most of the time, and once in a while, it chaotically enters a non-linear mode before settling down on another stable operating point that is offset by a multiple of T, and this corresponds to a cycle slip. At start-up, we try to locate the stable point at the origin and then in the tracking mode, we try to continue being in the linear region to prevent the occurrence of cycle slips. One method to do this is to improve the performance of the TED in the linear mode by operating closer to the origin a greater proportion of the time, thus reducing the probability of occurrence of cycle slips, otherwise called loss-of-lock.

Performance of the Mueller and Müller TED can be improved by using soft estimates $\tilde{d}_k$ in place of hard estimates $\hat{d}_k$. A good candidate for the soft-decision $\tilde{d}_k$ is the minimum-mean-squared error estimate of d_k given r_k, or equivalently $\tilde{d}_k = E[d_k|r_k]$. For the PR-IV channel, this leads to a memoryless soft slicer of the form

$$\tilde{d}_k = \frac{2\sinh(2r_k/\sigma^2)}{\cosh(2r_k/\sigma^2) + e^{2/\sigma^2}}. \tag{15}$$

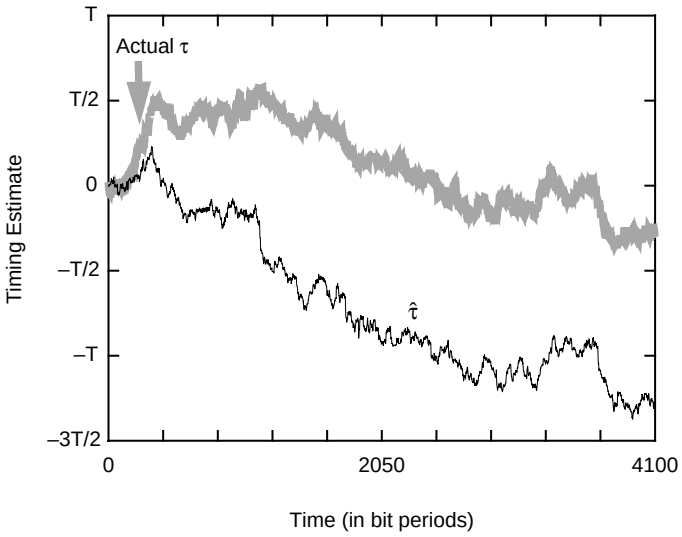

Figure 8: An example of a cycle slip.

The details of the derivation are in Appendix A. The normalized timing function for this *soft* MM TED is plotted in Figure 6 for SNR = 10 dB and denoted by "Soft". The timing function is almost identical to that of the hard decision case, but the error variance is lower. (See Figure 7.) The gap between the hard and the soft decision cases increases as the SNR decreases.

2.3 Conventional Coded System

So far, we have analyzed the timing recovery mechanisms for the uncoded system in some detail. However, practical systems employ equalization to combat channel distortions and also error control coding to combat errors introduced by the channel. In this section, we present some coding and equalization techniques used in conventional systems. In the Chapter 7, we present iterative timing recovery, which is an extension of the turbo principle to timing recovery, whereby timing recovery benefits from the presence of the error control code.

At the transmitter, the message bits to be transmitted to the receiver are first encoded using an error control encoder and then transmitted through a channel which

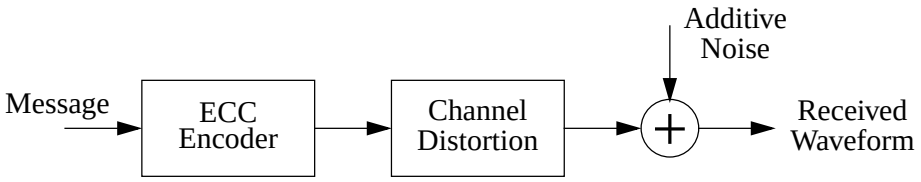

Figure 9: General block diagram of a coded system.

introduces distortion and additive noise, as shown in Figure 9. The receiver needs to undo the effects of both the channel (equalization) and the encoder (decoding) to recover the message bits.

We first discuss iterative error control codes which offer very low error rates [9] [25]. Next, we look at two different receiver architectures: one where equalization and decoding are performed sequentially; and another where equalization and decoding are performed jointly. The second architecture is called turbo equalization [54] and significantly outperforms the system where equalization and decoding are performed separately. In a conventional setting, the turbo equalization would be preceded by the timing recovery mechanism. In Chapter 7, we propose iterative timing recovery where turbo equalization and timing recovery are performed jointly, and this allows further improvements in performance.

Error control codes (ECCs), also called forward error-correction codes (FECs), are used to combat noise and errors introduced by the channel. Traditionally, ECCs have been categorized into block codes and convolutional codes. A new class of ECCs with iterative decoding techniques approach the Shannon channel capacity much closer than possible with conventional coding techniques. We consider two classes of iterative ECCs: turbo codes and low-density parity-check (LDPC) codes.

2.3.1 Turbo Codes

Turbo codes [9] are concatenated convolutional codes. The key to the good performance of turbo codes lies in the decoding algorithm. For the concatenated system,

instead of decoding the constituent codes independently, the decoders exchange information about the transmitted symbols and this improves performance significantly. First, we start with a brief description of the encoding and decoding of convolutional codes.

2.3.1.1 Convolutional Codes

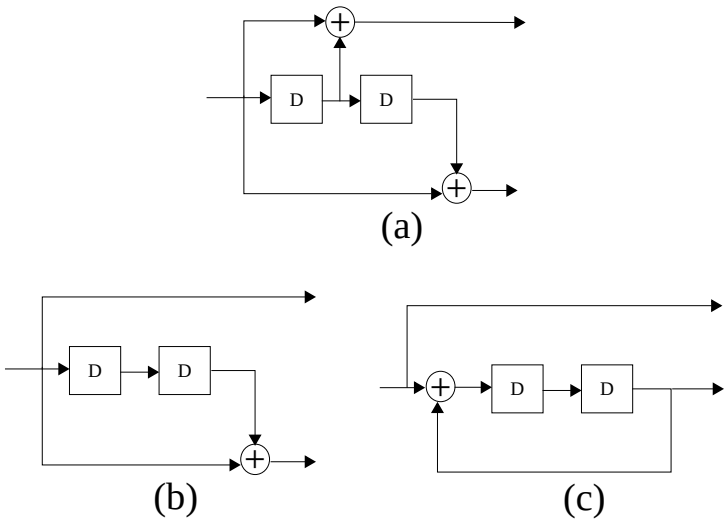

Figure 10: (a) A convolutional encoder; (b) a systematic convolutional encoder; and (c) a recursive systematic convolutional (RSC) encoder.

Error control codes work by introducing effective redundancy to the bit stream to be encoded. Convolutional codes use linear shift registers to do so [67]. The convolutional encoder acts on the input message stream and produces one or several output streams. The encoder consists of memory elements and modulo 2 adders. An example rate-1/2 encoder is shown in Figure 10(a). The memory elements are denoted by the symbol D. Convolutional encoders can be conveniently represented using the polynomial notation. An encoder is denoted by the polynomial $\sum_i a_i D^i$, where $a_i = 1$ if the i-delayed version of the input stream contributes to the output. For example, the encoder of Figure 10(a) is denoted by the pair $[1 \oplus D, 1 \oplus D^2]$. A systematic encoder is one where the input message stream appears intact at the

output. Figure 10(b) is an example of a systematic encoder. A recursive encoder is one where we have feedback, an example of which is in Figure 10(c). Figure 10(c) is, in fact, an example of a recursive systematic convolutional (RSC) encoder characterized by the polynomial pair $[1, 1/(1 \oplus D^2)]$.

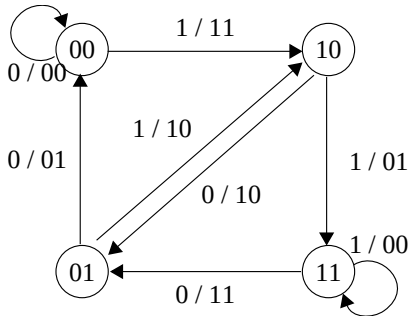

Figure 11: An example state diagram.

The analysis of convolutional codes is facilitated through the use of state diagrams. The contents of the memory elements constitute the state of the convolutional encoder and the encoding process can be thought of as a walk on the state diagram. The state diagram corresponding to the the example in Figure 10(a) is shown in Figure 11. Since we have two memory elements, we have four possible states denoted by 00, 01, 10 and 11 corresponding to the memory contents read from left to right. Possible transitions are shown by edges and the values (x/y_1y_2) along the edges are the input x and the outputs y_1 and y_2 from the top and the bottom branches respectively corresponding to that transition.

A trellis diagram is an alternative representation of the convolutional encoder that explicitly shows the passage of time. For example, a part of the trellis diagram corresponding to the state diagram of Figure 11 is shown in Figure 12. All the possible states of the system at any time instant k are plotted vertically. The k^{th} stage of the trellis represents all possible valid transitions of the encoder from a state at time instant k to another state at instant $k + 1$. The complete trellis is formed by replicating this stage as many times as needed. The branches (edges) are labeled in

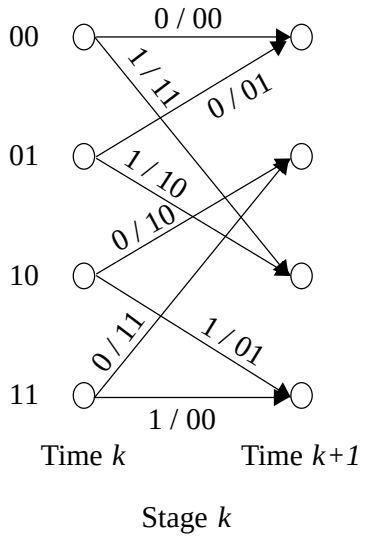

Figure 12: An example trellis diagram.

the same way as in the state diagram. A path through a trellis is a connected set of branches such that there is one branch for every stage. Every codeword is associated with a unique path through the trellis.

Convolutional codes have a linear-time decoder based on the Viterbi algorithm [23], which is a maximum-likelihood sequence estimator based on the trellis of the code. The outputs of the Viterbi decoder are hard decisions on the transmitted symbols. When convolutional codes are used as constituent codes in turbo codes, we need soft information about the transmitted bits and this is got by performing decoding based on the BCJR algorithm [3] as opposed to the Viterbi algorithm. The BCJR algorithm operates on the same trellis as the Viterbi algorithm and performs the maximum *a posteriori* estimation of the transmitted bits and outputs the *a posteriori* probability density associated with each code bit. The Viterbi algorithm outputs the most likely sequence of symbols whereas hard quantizing the output of the BCJR algorithm gives the sequence of most likely symbols.

2.3.1.2 The BCJR Algorithm

In this section, we present a summary of the BCJR algorithm [3] [5]. Let $\boldsymbol{a} = [a_0,\ a_1,\ \cdots\ a_{K-1}]$ be the message transmitted, and let $\boldsymbol{c} = [c_0,\ c_1,\ \cdots\ c_{N-1}]$ be the codeword. We assume that the message symbols are drawn from an alphabet $\mathcal{A}$. Let μ be the memory of the encoder. Let $\boldsymbol{r} = [\boldsymbol{r}_0,\ \boldsymbol{r}_1,\ \cdots\ \boldsymbol{r}_{K+\mu-1}]$ be the observation vector at the receiver where we have grouped the observations into $K+\mu$ group corresponding to the $K+\mu$ stages of the trellis. Since this is a rate K/N code, each message bit produces N/K (assuming it is an integer) code bits and therefore, N/K observations correspond to a trellis stage. This procedure can be modified suitably for the case where N/K is not an integer by having each stage correspond to K message bits and N code bits.

The MAP detector is the one that picks a to maximize the *a posteriori* probability $Pr[a_k = a|\boldsymbol{r}]$ for each k, where $a \in \mathcal{A}$. Let S_a denote the set of pairs (p, q) such that a branch from a state p to a state q corresponds to an input symbol a. Let ψ_k be the state of the encoder at time k. Then, we can rewrite the *a posteriori* probability as

$$Pr[a_k = a|\boldsymbol{r}] = \sum_{(p,q)\in S_a} Pr[\psi_k = p; \psi_{k+1} = q|\boldsymbol{r}]. \tag{16}$$

In effect, we have rewritten the symbol *a posteriori* probability in terms of the *a posteriori* probabilities of the transitions.

Using the Markov property of the encoder, the BCJR algorithm splits this *a posteriori* transition probability into three factors as follows: [3]

$$Pr[\psi_k = p; \psi_{k+1} = q|\boldsymbol{r}] = p(\psi_k = p; \boldsymbol{r}_{l<k})\ p(\psi_{k+1} = q; r_k|\psi_k = p)\ p(\boldsymbol{r}_{l>k}|\psi_{k+1} = q)\ /\ p(\boldsymbol{r}), \tag{17}$$

where we have separated the observation vector $\boldsymbol{r}$ into three components $\boldsymbol{r}_{l<k}$, r_k and $\boldsymbol{r}_{l>k}$ corresponding to the past, the present and the future symbols respectively with respect to the current index k. Defining

$$\alpha_k(p) \quad = \quad p(\psi_k = p; \boldsymbol{r}_{l<k}),$$

$$\begin{aligned} \gamma_k(p,q) &= p(\psi_{k+1}=q; r_k|\psi_k=p) \quad \text{and} \\ \beta_{k+1}(q) &= p(\boldsymbol{r}_{l>k}|\psi_{k+1}=q), \end{aligned} \tag{18}$$

we get

$$Pr[\psi_k=p;\psi_{k+1}=q|\boldsymbol{r}] = \alpha_k(p) \ \times \ \gamma_k(p,q) \ \times \ \beta_{k+1}(q) \ / \ p(\boldsymbol{r}). \tag{19}$$

$\alpha_k(p)$ is a function of the state p at time k that depends only on the past observations, and $\beta_{k+1}(q)$ is a function of the state q at time $k+1$ that depends only on the future observations. $\gamma_k(p,q)$ is a probability measure associated with a branch connecting state p at time k and state q at time $k+1$, and depends only on the current observation. The trellis representation of the encoder allows a recursive computation of the of $\alpha_{k+1}(q)$ in terms of $\alpha_k(p)$ and $\gamma_k(p,q)$ as follows.

$$\alpha_{k+1}(q) = \sum_{p=0}^{Q-1} \alpha_k(p)\gamma_k(p,q), \tag{20}$$

where Q is the total number of states possible at any time instant. Similarly, we have the following recursion for $\beta_k(p)$ in terms of $\beta_{k+1}(q)$ and $\gamma_k(p,q)$.

$$\beta_k(p) = \sum_{q=0}^{Q-1} \beta_{k+1}(q)\gamma_k(p,q). \tag{21}$$

The branch metrics $\gamma_k(p,q)$ depend on the channel. Consider the AWGN channel with $r_k = c_k + n_k$, where the noise variables $\{n_k\}$ are *i.i.d.* zero-mean Gaussian random variables of variance σ^2. The branch metric in this case is

$$\gamma_k(p,q) = \frac{1}{2\pi\sigma^2} \exp\left\{-\frac{|r_k - c^{(p,q)}|^2}{2\sigma^2}\right\} \Pr[a_k = a^{(p,q)}], \tag{22}$$

where $a^{(p,q)}$ and $c^{(p,q)}$ are respectively the input and the output symbols associated with the transition from state p to state q. As opposed to the Viterbi algorithm, in the BCJR algorithm, we make two passes through the trellis. In the forward pass, the values of $\alpha_k(p)$ are computed, and in the backward pass, the values of $\beta_k(q)$.

The outputs of the BCJR algorithm, *i.e.*, the probability values $Pr[a_k = a|\boldsymbol{r}]$ for all k and for all $a \in \mathcal{A}$, constitute the desired soft information. When we are dealing

with the binary alphabet $\{-1, +1\}$ the soft information is usually expressed as the logarithm of the ratio of the two probabilities, called the log likelihood ratio (LLR). The LLR (λ) is

$$\begin{aligned} \lambda_k &= \log \frac{Pr[a_k = 1|\boldsymbol{r}]}{Pr[a_k = -1|\boldsymbol{r}]}, \\ &= \log \frac{\sum_{(p,q)\in S_1} \alpha_k(p)\ \gamma_k(p,q)\ \beta_{k+1}(q)}{\sum_{(p,q)\in S_{-1}} \alpha_k(p)\ \gamma_k(p,q)\ \beta_{k+1}(q)}. \end{aligned} \tag{23}$$

2.3.1.3 Parallel Turbo Codes

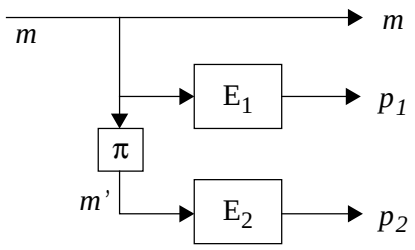

Figure 13: A parallel turbo encoder.

The encoder of a parallel turbo code is a parallel concatenation of two systematic recursive convolutional encoders (E_1 and E_2). Since both the encoders are systematic, one of the systematic branches can be dropped. Also, the message stream m is interleaved to get m', which serves as the input to one of the recursive encoders. Thus, at the encoder output, we have three streams: the message stream m and the two parity streams p_1 and p_2 from E_1 and E_2 respectively. This structure is shown in Figure 13. This is an example of a rate-1/3 parallel turbo code. The three streams are then interleaved and transmitted over the channel. At the receiver, the observations are separated into three streams r_m, r_1 and r_2 corresponding to m, p_1 and p_2 respectively. The receiver consists of two soft-input soft-output decoder blocks (D_1 and D_2) based on the BCJR algorithm. The decoder D_1 corresponds to the encoder E_1 and D_2 corresponds to E_2. The important innovation in decoding for the turbo codes is the fact that the decoders perform the decoding operation cooperatively by

exchanging soft information.

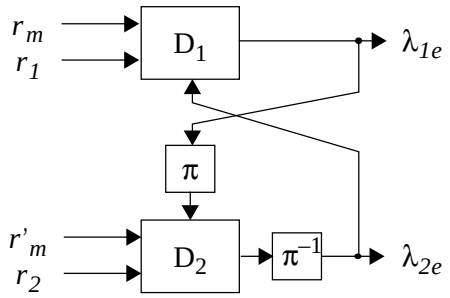

Figure 14: A parallel turbo decoder.

At the start of turbo decoding, D_1 acts on r_m and r_1 to produce soft information λ_1 about the message bits. The BCJR block in this case does not have any *a priori* information about the transmitted symbols. Then, this soft information λ_1 is fed to D_2 as *a priori* information when it implements the BCJR algorithm using the interleaved stream r'_m (interleaved to match with the interleaved message stream m') and r_2. Thus, D_2 utilizes the information already gleaned by D_1. Let λ_2 be the LLRs produced by D_2. To ensure that the feedback loop thus formed is stable, we pass only extrinsic information to D_1. Extrinsic information is the information extracted by the decoder, and is got by subtracting the decoder *a priori* input from the decoder output. Denote the de-interleaved extrinsic information output of D_2 by λ_{2e}. λ_{2e} is passed to D_1 as *a priori* information. Similarly, the *a priori* information for D_2 is an interleaved version of λ_{1e}, the extrinsic output of D_1. This turbo decoder structure is shown in Figure 14.

As iterations progress, the quality of the *a priori* information available to D_1 and D_2 improves, thus improving the quality of the output as well. At the end of turbo decoding, the final output is in the form of log-likelihood ratios of the form $\lambda_k = \lambda_{1e,k} + \lambda_{2e,k} + \lambda_{\text{sys},k}$, where λ_{sys} is the information got directly from the systematic portion of the observation vector. In other words, $\lambda_{\text{sys},k} = \log \frac{f(r_k|a_k=1)}{f(r_k|a_k=0)}$, where $f(\cdot)$ denotes the *p.d.f.* of the observation r_k conditioned on the bit a_k.

2.3.1.4 Serial Turbo Codes

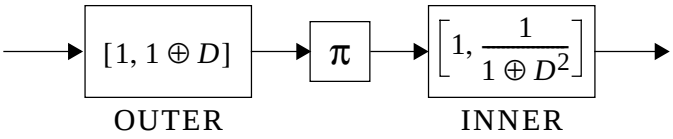

Figure 15: A rate-1/4 serial turbo encoder.

A serial turbo code is a serial concatenation of two convolutional encoders separated by an interleaver [61]. An example encoder is shown in Figure 15. This is an example of a rate-1/4 code with both the outer and the inner codes being of rate-1/2. The encoders are denoted by the polynomials used to generate the code bits. The outer encoder is a systematic encoder where the parity bits are generated according to $1 \oplus D^2$. For good performance, the inner code has to be recursive [7]. In the example of Figure 15, the inner encoder is a rate-1/2 systematic recursive encoder where the feedback path is $1 \oplus D^2$ and the parity bits are generated according to $1/(1 \oplus D^2)$.

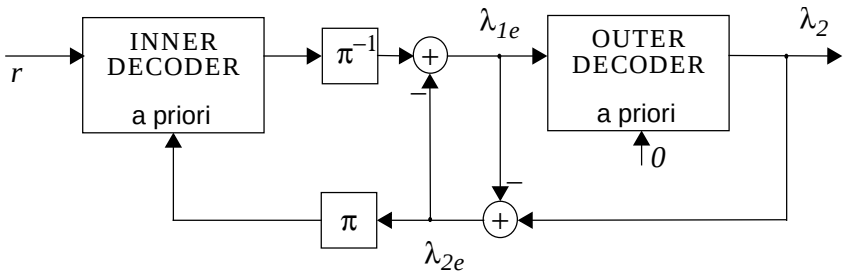

Figure 16: A serial turbo decoder.

The decoder structure is similar to that of a parallel turbo code in that the soft-in soft-out decoders corresponding to the two codes exchange soft extrinsic information. An important difference is that the outer decoder has to now produce soft information about its coded bits and not just the message bits [7]. Fortunately, this is simple enough to do by changing the numerator and denominator sets in the log likelihood evaluation. Instead of using the sets S_1 and S_{-1} corresponding to the message bit

being 1 or -1, we now use sets corresponding to the coded bit of interest being 1 or -1. The inner decoder operates on the channel observations and its extrinsic output is λ_{1e}. λ_{1e} is the input to the outer decoder, whose output is denoted by λ_2. The outer decoder assumes that the extrinsic output of inner decoder is the channel LLR. The *a priori* information for the outer decoder is always set to zero. At the end of turbo decoding, the systematic portion of λ_2 is the desired output. The decoder structure is shown in Figure 16.

2.3.2 Low Density Parity Check Codes

Turbo codes are based on convolutional codes. In this section, we discuss low-density parity-check (LDPC) codes [25], which are block codes. Block codes are characterized by a generator matrix $\boldsymbol{G}$ of dimensions $k \times n$ where k is the message length and n is the codeword length [67]. The codeword c of dimensions $1 \times n$ corresponding to a message m of dimensions $1 \times k$ is arrived at by the following operation.

$$c = m\boldsymbol{G}. \tag{24}$$

As before, if the message m appears intact in the codeword c, the code is a systematic code. In this case, after column permutations, the code matrix $\boldsymbol{G}$ can be rewritten as

$$\boldsymbol{G} = [\boldsymbol{I}_{k\times k}|\boldsymbol{P}_{n-k\times k}]\,, \tag{25}$$

where the identity matrix corresponds to the systematic portion and the matrix $\boldsymbol{P}$ corresponds to the parity bits in the codeword. A block code can alternately be represented by a parity-check matrix that is formed by the row vectors in the null space of the rows of $\boldsymbol{G}$. The parity-check matrix, denoted by $\boldsymbol{H}$, is a $(n-k)\times n$ matrix and satisfies $\boldsymbol{H}\boldsymbol{G}^T = 0$. Therefore, for any codeword, we have $\boldsymbol{H}c^T = \boldsymbol{H}\boldsymbol{G}^T m^T = 0$. In other words, the rows of $\boldsymbol{H}$ represent the parity-check equations that the bits of c satisfy. If we have the generator matrix in the systematic form, then the parity-check

matrix $\boldsymbol{H}$ is given by

$$\boldsymbol{H} = \left[-\boldsymbol{P}^T | \boldsymbol{I}_{n-k \times n-k}\right]. \tag{26}$$

We assume binary codes with codeword entries and the entries in $\boldsymbol{H}$ being from the set $\{0, 1\}$, and the addition operation being modulo 2. The density of a matrix is the ratio of the number of 1s in the matrix to the total number of entries in the matrix. A low-density parity-check code is a code whose parity check matrix has a low density. A randomly chosen low-density parity-check matrix would have very few 1s in each row and therefore, each parity-check equation involves only a few code bits. In other words, LDPC codes are characterized by sparse parity-check matrices [25]. A (j, k)-regular LDPC matrix is one which has exactly k ones in each row, and exactly $j < k$ ones in each column.

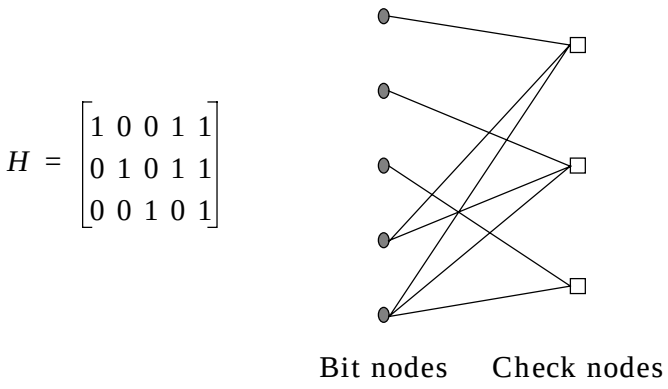

Figure 17: Tanner graph representation of a parity check matrix.

An alternative representation of a parity-check matrix is through bipartite graphs, also called Tanner graphs. A bipartite graph has two kinds of nodes: bit nodes, corresponding to the code bits; and check nodes, corresponding to parity-check equations, *i.e.*, the rows of $\boldsymbol{H}$. A bit node and a check node are connected with an edge if the code bit corresponding to the bit node participates in the parity-check equation corresponding to the check node. A sample parity-check matrix and its representation using a bipartite graph is shown in Figure 17.

The standard method of decoding block codes is through syndrome decoding [67]. The syndrome vector is defined to be $s = \boldsymbol{H}r^T$, where r is the received vector. The syndrome for a codeword evaluates to zero. The syndrome is chosen as an look-up index in a table of possible error vectors. The error vector corresponding to a syndrome is the most likely error vector that causes that syndrome, and is added to the received vector to get the receiver's decision. For practical code lengths, syndrome decoding is impractical due to the large number of possible error vectors. An attractive feature of LDPC codes is the existence of a much simpler decoding algorithm based on message-passing, which, assuming that the bipartite graph corresponding to the LDPC code is cycle-free, is optimal [25]. Even in the other cases where its optimality has not been proved, the message passing decoder performs remarkably well.

The message passing decoder works by passing messages between the bit nodes and the check nodes, improving the quality of its decisions at each iteration. This is similar to the decoding of turbo codes in that smaller entities within the decoder cooperate to decode the overall code. In fact, it has been shown that both turbo codes and LDPC codes are special cases of codes on generalized graphs with generalized trellis decoding algorithms [66].

2.3.2.1 Message Passing Decoder

In this section, we present a summary of the message passing algorithm [25] [5]. To simplify the notation, we present the message passing decoder assuming that we have a (j, k)-regular LDPC code. In the next section, we discuss design and decoding of irregular LDPC codes.

Let $\boldsymbol{a} = [a_0,\ a_1,\ \cdots\ a_{K-1}]$ be the message transmitted, where $a_i \in \mathcal{A}$ and $\mathcal{A}$ is the transmitter alphabet. The vector $\boldsymbol{a}$ is encoded to the codeword $\boldsymbol{c} = [c_0,\ c_1,\ \cdots\ c_{N-1}]$. Let $\boldsymbol{r} = [r_0,\ r_1,\ \ldots\ r_{N-1}]$ be the received vector. We again consider the MAP receiver

that picks a to maximize the *a posteriori* probability $Pr[a_k = a|\boldsymbol{r}]$ for each k, where $a \in \mathcal{A}$. For simplicity of notation, we assume that the alphabet $\mathcal{A}$ is binary with $\mathcal{A} = \{0, 1\}$. Therefore, the MAP receiver computes the *a posteriori* LLR λ_k given by

$$\lambda_k = \log \frac{Pr[a_k = 1|\boldsymbol{r}]}{Pr[a_k = 0|\boldsymbol{r}]}, \tag{27}$$

for each k and makes the decision $\hat{a}_k = 1$ if $\lambda_k > 0$, and $\hat{a}_k = 0$ if $\lambda_k <= 0$. Applying Bayes' rule and simplifying, we can express λ_k as

$$\lambda_k = \log \frac{f(r_k|a_k = 1)}{f(r_k|a_k = 0)} + \log \frac{\Pr[a_k = 1|\boldsymbol{r}_{i\neq k}]}{\Pr[a_k = 0|\boldsymbol{r}_{i\neq k}]}, \tag{28}$$

where $f(r_k|a_k = 1)$ is the *p.d.f* of the observation r_k given that $a_k = 1$. Similarly for $f(r_k|a_k = 0)$. The variable $\boldsymbol{r}_{i\neq k}$ is the vector that has all the observations r_i from $i = 0$ to $i = N - 1$ except for $i = k$. The first term in the summation on the right hand side represents the intrinsic information λ_k^{int} coming directly from the k^{th} channel observation, and the second term is the extrinsic information λ_k^{ext} for the k^{th} bit, coming from all the other channel observations. Message passing algorithm is a simple and effective way of computing extrinsic information.

Each of the bit nodes and the check nodes acts as a computation device that acts on the information received along the edges connected to it, and puts out information back onto these edges. Let the LLR values at the i^{th} iteration be denoted by $\{\lambda_k^i\}$. As iterations progress, assuming a cycle-free graph, λ_k^i converges to the true *a posteriori* LLR defined in (28). We initialize the LLR variables to the intrinsic values. Therefore,

$$\lambda_k^0 = \lambda_k^{int} \;=\; \log \frac{f(r_k|a_k = 1)}{f(r_k|a_k = 0)}. \tag{29}$$

Denote the set of indices of check nodes connected to bit node i by $M(i)$ and the set of indices of bit nodes connected to check node i by $N(i)$. Since we have a (j, k)-regular LDPC code, the cardinality of $M(i)$ is j for all i, and the cardinality of $N(i)$ is k for all i. Let the message from bit node i to the check node corresponding to the n^{th}

element in $M(i)$ be denoted by $m_{in}^{b\to c}$, and let the message from check node i to the bit node corresponding to the n^{th} element in $N(i)$ be $m_{in}^{c\to b}$.

First, consider the bit-nodes. Let l be the n^{th} element of $M(i)$. Therefore, check node l is the n^{th} check node connected to the i^{th} bit node. Let $\mathcal{M}_{il}^{c\to b}$ be the set of messages $m_{in}^{c\to b}$ for all $n \in M(i)\backslash\{l\}$, where $M(i)\backslash\{l\}$ denotes the set that contains all elements of $M(i)$ except for element l. The message from the i^{th} bit node to the l^{th} check node is

$$m_{in}^{b\to c} = \log \frac{\Pr[a_i = 1|\mathcal{M}_{il}^{c\to b}, \lambda_i^{int}]}{\Pr[a_i = 0|\mathcal{M}_{il}^{c\to b}, \lambda_i^{int}]}. \tag{30}$$

Therefore, the message $m_{in}^{b\to c}$ is the LLR computed at the i^{th} bit node taking into account inputs along all the edges except for the n^{th} one. Assuming a cycle-free graph, we get

$$m_{in}^{b\to c} = \lambda_i^{int} + \sum_{p\in M(i)\backslash\{l\}} m_{pq}^{c\to b}, \tag{31}$$

where the index q is such that the the q^{th} element in $N(p)$ is i. Therefore, the message from the i^{th} bit node to the n^{th} check node connected to it is simply the sum of the intrinsic information and all the incoming check-to-bit messages except from the check node in question. The check-to-bit messages are all initialized to 0. Therefore, the bit-to-check messages for the first iteration are

$$m_{in}^{b\to c} = \lambda_i^{int}, \tag{32}$$

for all i and $n \in M(i)$.

The LLR for the i^{th} bit on the kth iteration is given by

$$\lambda_i^k = \lambda_i^{int} + \sum_{p\in M(i)} m_{pq}^{c\to b}, \tag{33}$$

where the index q is such that the the q^{th} element in $N(p)$ is i. In other words, the LLR value is simply the sum of the intrinsic information and all the incoming messages.

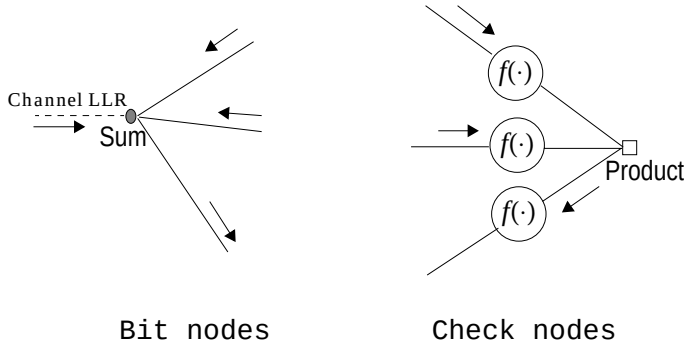

Figure 18: The bit and check updates in message passing decoding.

Next, the check node computations. Let l be the n^{th} element of $N(i)$. Therefore, bit node l is the n^{th} bit node connected to the i^{th} check node. Let $\mathcal{M}_{il}^{b\to c}$ be the set of messages $m_{in}^{b\to c}$ for all $n \in N(i)\backslash\{l\}$. The message from the i^{th} check node to the l^{th} bit node is

$$m_{in}^{c\to b} = \log \frac{\Pr[a_l = 1|\mathcal{M}_{il}^{b\to c}]}{\Pr[a_l = 0|\mathcal{M}_{il}^{b\to c}]}. \tag{34}$$

The check node computation is not as simple as a summation, which was the case with the bit nodes. For the check nodes, we get

$$m_{in}^{c\to b} = -2\tanh^{-1}\left(\prod_{p\in N(i)\backslash\{l\}} \tanh\left(\frac{-m_{pq}^{b\to c}}{2}\right)\right), \tag{35}$$

where the index q is such that the q^{th} element in $M(p)$ is i. Even at the check node, for any outgoing message, the operation is on all the incoming messages except on the edge of interest. But, instead of a summation, we have the $\tanh^{-1}$ of a product of tanh of the relevant incoming messages. The bit node and the check node operations are summarized in Figure 18, where the function $f(x)$ is defined by $f(x) = \tanh\left(-\frac{x}{2}\right)$.

The check-to-bit messages contain extrinsic information. Consider the i^{th} bit node at the first iteration. The incoming check-to-bit messages are from check nodes with indices in $M(i)$. Each of these is connected to $k-1$ bits other than the i^{th} bit node. Therefore, each of the incoming check-to-bit messages incident on the i^{th} bit has information about a_i based on some $k-1$ observations in the set $\boldsymbol{r}_{l\neq i}$. Under the

assumption of a cycle-free graph, we can say that, after the first iteration, we have information about a_i based on $j(k-1)$ out of the $N-1$ elements of $\boldsymbol{r}_{l\neq i}$. By a similar argument, we can show that at the end of the second iteration, we have information about a_i based on $j(j-1)(k-1)^2$ elements of $\boldsymbol{r}_{l\neq i}$. As iterations progress, eventually, the sum of check-to-bit messages at the i^{th} bit node has information about a_i based on all the elements of $\boldsymbol{r}_{l\neq i}$. This is nothing but the extrinsic information of (28).

When the graph has cycles, the message passing decoder no longer gives the exact MAP decoder, but is approximate. However, in practice, the performance of the message passing decoder is good even in presence of cycles.

2.3.2.2 Irregular LDPC Codes

Regular LDPC codes show good performance with the message passing decoder at reasonably low complexity, but to approach capacity, we need to use irregular codes [55]. As the name suggests, irregular LDPC codes are not regular. Not all the bit nodes have the same degree (number of edges connected to it) and not all the check nodes have the same degree. Irregular LDPC codes are specified using degree distribution polynomials, which are convenient ways of representing the fraction of nodes that are connected to a certain number of edges. The bit node degree distribution polynomial is $\rho(x) = \sum_i \rho_i x^i$, where ρ_i is the fraction of bit nodes that have degree i. The check node degree distribution polynomial is $\lambda(x) = \sum_i \lambda_i x^i$, where λ_i is the fraction of check nodes that have degree i. Irregular LDPC codes allow an additional degree of freedom when compared to regular ones, and this has been exploited to design codes that perform very close to the Shannon capacity [13].

The performance of LDPC codes can be predicted as a function of the bit node and the check node degree distributions using density evolution [14]. Density evolution tracks the probability density of the messages that are passed between the bit nodes and the check nodes. For high enough SNRs, the mean of the densities goes to

infinity as iterations progress, meaning that the decoder is sure of its decisions. For low enough SNRs, the mean converges to a finite value and this represents the case where the decoder fails to decode. The SNR that divides these two regimes represents the threshold for the performance of the code. Irregular LDPC codes are designed to get the threshold as close to the Shannon capacity as possible.

The decoding of irregular LDPC codes is using message passing as described in the previous section, except that in each of the evaluations, the number of participating messages is not fixed, but rather depends on the degree of the node in question.

2.3.3 Turbo Equalization

Equalization is used to combat the distortion introduced by the channel. Conventionally, linear equalizers and decision feedback equalizers have been used for this purpose. An alternate method is to view the channel as a rate-1 convolutional encoder that can be decoded on the trellis. This approach is feasible if the channel has few taps as this means we have a manageable number of states in the trellis. When this is not the case, we traditionally split the process of equalization into two stages: the first stage equalizes to a partial response target with few taps that is spectrally similar to the channel response; and in the second stage we decode using the trellis based on the target impulse response. The first stage minimizes noise enhancement and at the same time limits the number of taps for the next stage, and in stage two, we perform the maximum-likelihood or maximum *a posteriori* decoding. This is the basis of the partial response maximum likelihood (PRML) technique used in magnetic recording [15]. A widely-used target for the magnetic recording channel is the PR-IV channel, also characterized by the polynomial $1 - D^2$. The output of the PR-IV channel is got by subtracting the delayed-by-two version of the input from the input itself.

The turbo idea, where the constituent components of the decoded exchange information, has been extended to include equalization as well, and this is called turbo

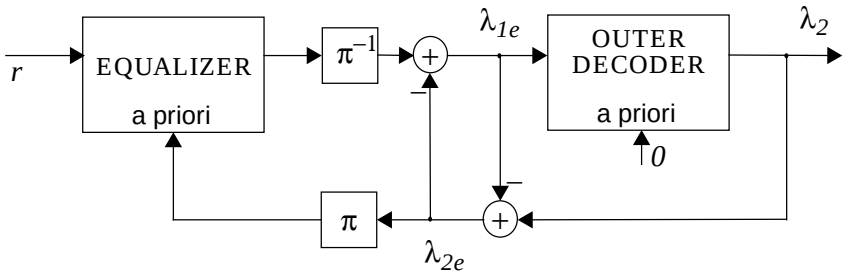

Figure 19: A turbo equalizer.

equalization [54]. The system now is viewed as a serial concatenated turbo code with the channel being the inner code and the ECC being the outer code. The decoder structure is exactly the same as that of a serial turbo code, except that the inner decoder is now replaced by the equalizer, as shown in Figure 19. An advantage with this scheme is that the ECC now need not be a turbo code to get good performance. A convolutional code used as the ECC gives surprisingly good performance with the $1/1 \oplus D^2$-precoded PR-IV channel, and this can be attributed to the precoded PR-IV channel being a good inner code for the serial turbo configuration. This is the configuration that will be studied in subsequent chapters. In addition, we also consider LDPC codes as the outer codes.

2.4 Iterative Synchronization

In this section, we summarize previous work related to iterative timing recovery. Iterative algorithms have been proposed in the past for phase synchronization, carrier synchronization and timing recovery for different channel models.

Perhaps the first such instance was the work of Georghiades and Snyder [26], where the expectation-maximization (EM) algorithm was used to perform sequence detection in the presence of a fixed timing offset and a simple channel model where the ISI for any symbol was only due to its immediate neighbors. In this case, the EM algorithm converged fast and gave good results.

2.4.1 AWGN Channel and Constant Offset

The most popular application of iterative techniques for synchronization appears to be in the area of carrier phase recovery. The most common system model employed is the one involving an AWGN channel and a carrier phase offset that remains constant for the duration of the packet. Either QAM or PSK constellations are used for the signal modulation. The ECC for the system is either a turbo code or an LDPC code. For this channel model, the timing estimator used is either the maximum-likelihood (ML) or the maximum *a posteriori* one. The timing estimator is used in conjunction with the decoding iterations of the ECC, with soft information from the ECC decoder being used in the timing estimator, and better samples due to the timing estimator being used in the ECC decoder.

Specific instances of the general framework described above are as follows. In [50], it is shown that the ML timing estimator can be implemented using the EM algorithm. The EM iterations are interpreted as the iterations of the turbo decoder with the timing estimator. The system considered here is an AWGN system with a PSK constellation. In [40], the system model considered is a rate-1/3 turbo-coded system on an AWGN channel with a QPSK constellation with phase offset and a timing offset, both of which are constant for the duration of the packet. The effect of timing errors on the BCJR algorithm is quantified, and this analysis is used to propose a new receiver architecture. In the turbo iterations, the proposed receiver replaces the BCJR block by a new MAP block which runs two BCJR blocks for two possible timing offsets, chosen according to available *a priori* information. In [41], a slightly different approach is taken to tackle the phase offset. The trellis of the turbo code is frequently terminated using regular pilot bits and tail bits. This adds redundancy, but improves the performance of the timing estimator and the receiver in general. In [70] and [71], for a BPSK system on an AWGN channel, an explicit maximum *a posteriori* phase estimator is derived and this is used as the timing estimator in

iterations with the turbo decoder.

Similar results and algorithms for QAM constellations are obtained in [34] and [35], where a constant phase offset is tracked on an AWGN channel. The timing estimators in these cases are based on the ML and a pseudo-ML criteria respectively. As before, the timing estimator uses soft information from the turbo iterations.

As opposed to using turbo codes as the ECC, the following contributions use an LDPC code as the outer code. In [62], the system model is the same as the one used in [35], except that the ECC is now an LDPC code. The receiver architecture consists of an ML timing estimator that iterates with an LDPC decoder. In [51], for an LDPC-coded system with BPSK symbols on an AWGN channel with a constant phase offset, an iterative receiver is proposed that works on the augmented factor graph that includes nodes for estimating the constant phase offset. The additional timing nodes implement a low-complexity heuristic algorithm that uses hard-quantized versions of the soft information as training symbols for phase estimation. The performance of this algorithm is analyzed using density evolution.

All the methods described above essentially feedback soft decisions from the ECC decoder to the synchronization block to improve its performance. A different approach to account for the presence of ECC in the system is described in [60]. Depending on the receiver's estimate ($\hat{p}$) of the probability of error (p), a non-linearity is used to modify the timing error signal in the PLL. The performance of the overall loop improves as a result of this modification. One way to arrive at the values of $\hat{p}$ is to use extrinsic information charts. For each iteration, a suitable $\hat{p}$ is used. The algorithm presented is also shown to be robust to mismatches between p and $\hat{p}$.

2.4.2 AWGN Channel and Time-varying Offset

So far, all the results mentioned have been for tracking a constant phase offset with AWGN channels. A more general system model would have time-varying synchronization parameters. In [42], a rate-1/3 turbo-coded, BPSK-encoded system is used on an AWGN channel with severe phase noise modeled by a Markov chain. To simplify the system implementation, the possible phase offsets are quantized to a certain number of levels. The BCJR algorithm is then modified to increase the state-space to include the effects of the phase offsets, leading to good performance even in the face of severe phase noise. An alternative approach is based on per-survivor processing (PSP), where the phase estimation and data detection are jointly performed by having multiple phase estimators running in parallel, one for each state of the trellis. This is explored for the AWGN channel with severe phase noise for a very low-rate turbo-coded system in [27]. Phase estimation for an M-PSK system faced with a random walk phase offset is considered in [11]. Here too, the solution proposed is an EM-based estimator that uses per-survivor processing.

For higher constellations like QPSK and 8-PSK, the modified BCJR algorithm of [42] becomes rather complex. As an alternative, in [22], a random walk phase offset is tracked using a first-order PLL in a per-survivor processing (PSP) fashion, with multiple PLLs running for the different states of the SISO decoders of the turbo decoder. In [52], stochastic gradient descent type algorithms are used to track a frequency offset. These take the form of first and second order update equations. Phase estimation and derotation are performed between successive turbo decoder iterations.

A common method for generating soft decisions for a turbo-coded system is using the BCJR algorithm. To reduce complexity, in [43], a new method to generate soft decisions has been proposed. This method relies on partitioning the appropriate parameter space, and leads to a fast suboptimal algorithm for phase estimation for

turbo-coded, QAM-encoded systems on an AWGN channel. This method is general and is applicable to both constant phase offset and random walk phase offset.

As with the constant phase offset case, LDPC codes have been used as the outer code with time-varying phase offset models. In [33], phase estimation is performed for an LDPC-coded BPSK system with no ISI using a generalized Forney factor graph that has additional nodes to account for the phase offset. These additional nodes implement gradient methods or Kalman filtering using quantized messages.

2.4.3 ISI Channel and Time-varying Offset

The next level of generalization that we consider is the inclusion of an ISI channel. All the work mentioned so far has been for single-user AWGN channels. In [18], the channel model considered is multi-user AWGN. For this setting, chip-level synchronization is achieved using an iterative synchronization algorithm that implements local grid search. Calabro-Wolf perfect arrays are used to aid the synchronization process on a channel dedicated to synchronization.

In [65], the problem of timing recovery is considered for ISI-channels that employ turbo codes. The timing offset considered here is constant for the duration of the packet, and the EM algorithm is used. Simulation results with square-root raised-cosine pulse shape show that the proposed estimator performs close to one that achieves the Cramér-Rao bound.

Timing recovery for ISI channels, specifically partial response (PR) channels, that suffer from time-varying offsets is considered in [69]. A novel MAP detector is presented that includes the effects of ISI and the quantized timing offsets. As opposed to the case with AWGN channels where the synchronization and decoding was performed jointly, with ISI channels, there is the added requirement of equalization. Ideally, the three tasks of synchronization, equalization and decoding need to be performed jointly. However, this is prohibitively complex. In [69] and [4], synchronization and

equalization are jointly performed using the proposed MAP detector and this iterated with the ECC decoder. An alternate approach is presented in [31], where the PSP-based BCJR method is used to jointly perform synchronization and equalization. In this paper, multiple PLLs are employed to generate different samples for each of the states of the BCJR while tracking pseudo-survivor sequences. As before, soft information from the ECC decoder is used in the synchronization-equalization block. Two versions, symbol-rate and oversampled, are presented, with the oversampled version outperforming the symbol-rate one.

The use of cycle-slip detector is discussed in [30]. The channel considered in a partial response channel. The cycle-slip detector is based on soft information from the ECC decoder, and for a slowly-varying sinusoidal timing offset, the cycle-slip detector is shown to appreciably reduce the number of iterations needed.

Another type of channel considered is the fading channel common in wireless environments. In [72], a continuous phase modulation (CPM) system is considered in a flat-fading channel with AWGN and a time-varying phase offset. The MAP estimation algorithm proposed takes the form of a forward-backward algorithm with multiple PLLs used based on the PSP principle. For a time-varying channel, it is shown that it is beneficial to run an approximate forward-only algorithm. In [68], a multi-user DS-CDMA setting with a single-parameter timing offset is considered in a fading environment. Pilot symbols are used during the synchronization phase, thus obviating the need for iterations with the ECC decoder. Synchronization and channel estimation are performed jointly, with the timing estimates being arrived at by gradient search based on the channel estimates and the pilot symbols.

Finally, our work on iterative timing recovery fits in with the work presented in this subsection, namely timing recovery for ISI channels that suffer from time-varying timing offsets. We consider systems that have baud-rate sampling. We approximate the optimal process of joint synchronization, equalization and decoding by iterating

between the synchronization, equalization and decoding blocks. We use the simple PLL-based timing recovery block, and the process of iteration gets us close to the performance of a system with perfect synchronization. We also use slope-based cycle-slip detectors to speed up convergence.

2.5 Summary

We presented a brief overview of the magnetic recording channel. In this work, we are concerned with timing recovery for ISI-limited channels that allow baud-spaced sampling, and we use the magnetic recording channel as an example. To ensure low error rates in a typical magnetic recording system, we need to undo non-idealities introduced by the channel, and to this effect we perform timing recovery, equalization, precoding, RLL encoding and decoding, and ECC encoding and decoding.

Next, we set up the timing recovery problem and described the conventional timing recovery method for the magnetic recording channel. This is based on a PLL which uses timing error estimates from a TED to update its timing estimates. We also proposed the soft TED that uses soft decisions instead of hard decisions. The proposed soft TED has lesser measurement error variance than the hard TED. Finally, we presented a broad overview of previous and current work on iterative synchronization.

While designing timing recovery algorithms, the system is assumed to be uncoded, which leads to the assumption of *i.i.d.* data symbols. Instantaneous decisions about the transmitted symbols are used in the decision-directed TED, assuming that these decisions are correct. These assumptions are valid for systems operating at high SNR. The advent of capacity-approaching codes allows operations at low SNR where the instantaneous decisions are not reliable enough. The assumption of independent data symbols, however, is still approximately valid with capacity-approaching codes. Therefore, while deriving the Cramér-Rao for the timing recovery problem in Chapter 3, we assume that the data symbols are independent and identically distributed.

CHAPTER 3

CRAMÉR-RAO BOUND

In Chapter 2, we presented a synopsis of the magnetic recording system under consideration and also of conventional timing recovery. In this chapter, we consider the timing recovery problem for a more general ISI-limited system that allows baud rate sampling. Using the Cramér-Rao bound (CRB) [63] as a basis, we develop lower bounds on the timing estimation error variance of any *unbiased* estimator. For tractability we assume an uncoded system with *i.i.d.* data symbols, which is approximately true for the capacity-approaching iteratively-decodable codes considered in our study.

The bounds developed in this chapter are used in Chapter 4 to evaluate the performance of the conventional PLL-based timing recovery methods. In Chapter 5, we develop block-processing timing recovery methods that outperform the conventional ones. In addition to serving as a benchmark to evaluate the different timing recovery schemes of Chapters 4 and 5, the CRB evaluation allows an analytical approximation to the MAP estimator of Chapter 5 and also leads to a much improved acquisition strategy in Chapter 6.

The structure of the rest of the chapter is as follows. In Section 3.1, we define the Cramér-Rao bound (CRB) and arrive at the form of the CRB that is used in our derivations. Next, we present the system model under consideration in Section 3.2. We then derive the CRB for this system for different timing models in Section 3.3 and finally, we present the summary of the results and observations in Section 3.4.

3.1 Definition

Parameter estimation problems can be modeled as consisting of the following four components: [63]

- *Parameter Space.* The parameter $\boldsymbol{\theta}$ is a $P \times 1$ vector consisting of P scalar parameters. The parameter space is the set of all possible values the parameter may take on.

- *Observation Space.* This is the set of all possible observations $\boldsymbol{r}$. In general, $\boldsymbol{r}$ is a vector of dimensions $N_r \times 1$.

- *Probabilistic Mapping from Parameter Space to Observation Space.* This is the conditional probability density $f_{\boldsymbol{r}|\boldsymbol{\theta}}(\boldsymbol{r}|\boldsymbol{\theta})$ that determines the effect of the parameter $\boldsymbol{\theta}$ on the observation $\boldsymbol{r}$.

- *Estimation Rule.* This is the mapping rule $\hat{\boldsymbol{\theta}}(\boldsymbol{r})$ that we use to map the observation $\boldsymbol{r}$ into an estimate $\hat{\boldsymbol{\theta}}$ which is again a $P \times 1$ vector.

In the absence of *a priori* information about the parameter $\boldsymbol{\theta}$, we treat it as a deterministic but unknown quantity. When we have an *a priori* distribution $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$, we treat $\boldsymbol{\theta}$ as a random variable.

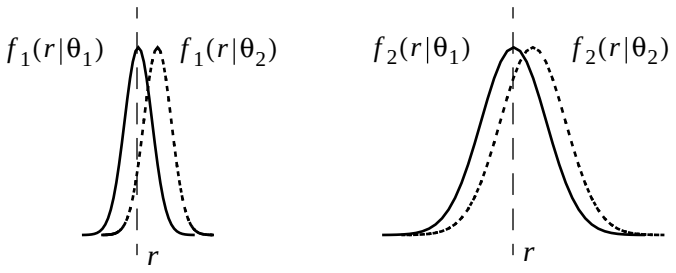

Figure 20: CRB depends on the *narrowness* of the conditional probability density.

Example: Before the precise definition of the CRB, we first present the following intuition about the CRB. Let θ be the parameter to be

estimated. In Figure 20, we consider two example density functions $f_1(r|\theta) = \mathcal{N}(\theta, \sigma_1^2)$ and $f_2(r|\theta) = \mathcal{N}(\theta, \sigma_2^2)$, where $\sigma_1^2 < \sigma_2^2$. Let θ_1 be the actual realization of θ, and let θ_2 be the estimate. For any particular realization of the observation r, we see that the change in $f_1(r|\theta)$ going from $\theta = \theta_1$ to $\theta = \theta_2$ is greater than the change in $f_2(r|\theta)$ going from $\theta = \theta_1$ to $\theta = \theta_2$. In other words, with $f_1(r|\theta)$, the likelihood of observations leading away from θ is less that those with $f_2(r|\theta)$, and hence we can expect lesser error variance for estimates of the parameter θ with $f_1(r|\theta)$ than with $f_2(r|\theta)$. Therefore, the sensitivity of $f(r|\theta)$ to changes in θ determines the quality of the estimate. The narrower the conditional probability density, the better the estimate. In the absence of *a priori* information, the CRB uses $\frac{\partial}{\partial\theta} \ln f(r|\theta)$ as the measure of narrowness. With *a priori* information, the measure of narrowness is $\frac{\partial}{\partial\theta} \ln f(r, \theta)$.

First, consider the case without *a priori* information. We are interested in the estimation error covariance matrix $E_{\boldsymbol{r}}[(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})^T]$. An estimator $\hat{\boldsymbol{\theta}}(\boldsymbol{r})$ is called *unbiased* if, for all $\boldsymbol{\theta}$, the average bias $B(\boldsymbol{\theta})$ is zero, where

$$B(\boldsymbol{\theta}) = E[\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta}] \triangleq \int_{-\infty}^{\infty} f_{\boldsymbol{r}|\boldsymbol{\theta}}(\boldsymbol{r}|\boldsymbol{\theta})[\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta}] \, \mathrm{d}\boldsymbol{r}. \tag{36}$$

For any *unbiased* estimator, the CRB is given by [63]

$$E_{\boldsymbol{r}}[(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})^T] \geq \boldsymbol{J}_\theta^{-1}, \tag{37}$$

where $\boldsymbol{J}_\theta$ is a $P \times P$ square matrix called the Fisher information matrix, defined by

$$\boldsymbol{J}_\theta = E_{\boldsymbol{r}} \left\{ \left[\frac{\partial}{\partial \boldsymbol{\theta}} \ln f_{\boldsymbol{r}|\boldsymbol{\theta}}(\boldsymbol{r}|\boldsymbol{\theta}) \right] \left[\frac{\partial}{\partial \boldsymbol{\theta}} \ln f_{\boldsymbol{r}|\boldsymbol{\theta}}(\boldsymbol{r}|\boldsymbol{\theta}) \right]^T \right\}. \tag{38}$$

The matrix inequality $E_{\boldsymbol{r}}[(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})^T] \geq \boldsymbol{J}_\theta^{-1}$ is in the sense that the matrix difference $\left(E_{\boldsymbol{r}}[(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})^T] - \boldsymbol{J}_\theta^{-1} \right)$ is positive semi-definite. This implies that the estimation error variance for the i^{th} parameter θ_i is bounded by the

i^{th} diagonal entry of $\boldsymbol{J}_\theta^{-1}$. In symbols,

$$E_{\boldsymbol{r}}[(\hat{\theta}_i(\boldsymbol{r}) - \theta_i)^2] \geq \boldsymbol{J}_\theta^{-1}(i,i), \tag{39}$$

where $\hat{\theta}_i(\boldsymbol{r})$ and θ_i denote the i^{th} element of $\hat{\boldsymbol{\theta}}(\boldsymbol{r})$ and $\boldsymbol{\theta}$ respectively.

In the presence of an *a priori* distribution $f_{\boldsymbol{\theta}}(\boldsymbol{\theta})$, we consider the average error covariance matrix $E_{\boldsymbol{r},\boldsymbol{\theta}}[(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})^T]$. As opposed to *unbiased* estimators, we consider *probabilistically unbiased* estimators where the conditional expectation of the error, $B(\boldsymbol{\theta})$, satisfies the following conditions:

$$\begin{aligned} \lim_{\boldsymbol{\theta}\to\boldsymbol{\infty}} B(\boldsymbol{\theta}) f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) &= \mathbf{0}, \\ \lim_{\boldsymbol{\theta}\to-\boldsymbol{\infty}} B(\boldsymbol{\theta}) f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) &= \mathbf{0}. \end{aligned} \tag{40}$$

The limits in the above equations are in the sense that each individual component tends to ∞ or $-\infty$ respectively as the case may be. An *unbiased* estimator is *probabilistically unbiased* and, therefore, the result below holds for a more general class of estimators. For *probabilistically unbiased* estimators, the lower bound is given by [63]

$$E_{\boldsymbol{r},\boldsymbol{\theta}}[(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})(\hat{\boldsymbol{\theta}}(\boldsymbol{r}) - \boldsymbol{\theta})^T] \geq \boldsymbol{J}_T^{-1}, \tag{41}$$

where $\boldsymbol{J}_T$ is the total information matrix of dimensions $P \times P$, defined by

$$\boldsymbol{J}_T = E_{\boldsymbol{r},\boldsymbol{\theta}} \left\{ \left[\frac{\partial}{\partial \boldsymbol{\theta}} \ln f_{\boldsymbol{r},\boldsymbol{\theta}}(\boldsymbol{r},\boldsymbol{\theta}) \right] \left[\frac{\partial}{\partial \boldsymbol{\theta}} \ln f_{\boldsymbol{r},\boldsymbol{\theta}}(\boldsymbol{r},\boldsymbol{\theta}) \right]^T \right\}, \tag{42}$$

where $f_{\boldsymbol{r},\boldsymbol{\theta}}(\boldsymbol{r},\boldsymbol{\theta})$ is the joint probability density of $\boldsymbol{r}$ and $\boldsymbol{\theta}$. As before, the matrix inequality is in the sense that the difference is positive semi-definite. Note that $\boldsymbol{J}_T$ can be rewritten as

$$\boldsymbol{J}_T = E_{\boldsymbol{\theta}}[\boldsymbol{J}_\theta] + \boldsymbol{J}_a, \tag{43}$$

where the *a priori* information matrix $\boldsymbol{J}_a$ is

$$\boldsymbol{J}_a = E_{\boldsymbol{r},\boldsymbol{\theta}} \left\{ \left[\frac{\partial}{\partial \boldsymbol{\theta}} \ln f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \right] \left[\frac{\partial}{\partial \boldsymbol{\theta}} \ln f_{\boldsymbol{\theta}}(\boldsymbol{\theta}) \right]^T \right\}. \tag{44}$$

The Fisher information matrix $\boldsymbol{J}_\theta$ has an elegant form when the density $f_{\boldsymbol{r}|\boldsymbol{\theta}}(\boldsymbol{r}|\boldsymbol{\theta})$ is multivariate normal [59]. Specifically, if the distribution of $\boldsymbol{r}$ is $\mathcal{N}(\boldsymbol{x}(\boldsymbol{\theta}), \boldsymbol{R})$, *i.e.*,

$$f_{\boldsymbol{r}|\boldsymbol{\theta}}(\boldsymbol{r}|\boldsymbol{\theta}) = \frac{1}{\sqrt{(2\pi)^N |\boldsymbol{R}|}} \exp\left(-\frac{1}{2}\left[\boldsymbol{r} - \boldsymbol{x}(\boldsymbol{\theta})\right]^T \boldsymbol{R}^{-1}\left[\boldsymbol{r} - \boldsymbol{x}(\boldsymbol{\theta})\right]\right), \tag{45}$$

then $\boldsymbol{J}_\theta$ can be simplified to

$$\boldsymbol{J}_\theta = \boldsymbol{G}^T \boldsymbol{R}^{-1} \boldsymbol{G}, \tag{46}$$

where $\boldsymbol{G}$ is the *sensitivity matrix*

$$\boldsymbol{G} = \left[\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{x}^T(\boldsymbol{\theta})\right] = \left[\boldsymbol{g}_0\ \boldsymbol{g}_1 \cdots \boldsymbol{g}_{P-1}\right], \tag{47}$$

P is the size of the vector $\boldsymbol{\theta}$ and the $N \times 1$ column vector $\boldsymbol{g}_i = \frac{\partial \boldsymbol{x}(\boldsymbol{\theta})}{\partial \theta_i}$ denotes the variation in the mean $\boldsymbol{x}(\boldsymbol{\theta})$ with respect to θ_i, the i^{th} element of $\boldsymbol{\theta}$. When the covariance matrix is of the form $\boldsymbol{R} = \sigma^2 \boldsymbol{I}$, this further simplifies to

$$\boldsymbol{J}_\theta = \frac{1}{\sigma^2} \boldsymbol{G}^T \boldsymbol{G}. \tag{48}$$

This is the form that shall be used in the sequel.

3.2 System Model

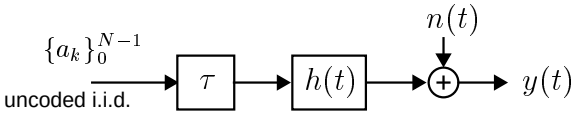

Figure 21: System block diagram with timing offsets, channel distortion and additive noise.

Consider the pulse amplitude modulated (PAM) system shown in Figure 21. The channel output waveform $y(t)$ is given by

$$y(t) = \sum_{l=0}^{N-1} a_l h(t - lT - \tau_l) + n(t), \tag{49}$$

where T is the bit period, $a_l \in \{\pm 1\}$ are the N *i.i.d.* data symbols, $h(t)$ is the channel impulse response, $n(t)$ is additive white Gaussian noise, and τ_l is the unknown timing offset for the l^{th} symbol.

We assume that the channel response $h(t)$ is band-limited to the frequency range $[-\frac{1}{2T}, \frac{1}{2T})$ with the result that baud-rate samples taken at a sampling rate of $1/T$ provide sufficient statistics. To eliminate out-of-band noise at the receiver, we filter the received waveform $y(t)$ by a low-pass filter with impulse response $\frac{1}{T}\frac{\sin(\pi t/T)}{\pi t/T}$. Next, we sample the resulting waveform $r(t)$ at instants kT to arrive at baud-rate samples $\{r_k\}$ given by

$$r_k = \sum_{l=0}^{N-1} a_l h(kT - lT - \tau_l) + n_k, \tag{50}$$

where n_k are zero-mean *i.i.d.* normal random variables with variance σ^2. We define the signal-to-noise ratio (SNR) to be $\text{SNR} = \frac{E_h}{2\sigma^2}$ where $E_h = \int_{-\infty}^{\infty} |h(t)|^2 \, \mathrm{d}t$ is the energy in the channel impulse response $h(t)$. We assume that we collect $N+2M$ samples and stack them in a observation vector $\boldsymbol{r} = [r_{-M} \; r_{-M+1} \; \ldots \; r_{N+M-1}]^T$, where we eventually let $M \to \infty$.

The receiver has to estimate the symbols $\{a_k\}$ and the timing offsets $\{\tau_k\}$ in the general case. In the trained case, where the transmitted symbols $\{a_k\}$ are known at the receiver, the problem is simpler and the receiver only needs to estimate $\{\tau_k\}$. The general timing recovery problem can be phrased as one where we need to estimate $\{a_k\}$ and $\{\tau_k\}$ given $\{r_k\}$ and knowledge of the channel response $h(t)$.

3.3 Evaluation of the CRB

In this section, we evaluate the CRB for four timing offset models of interest, assuming that the transmitted symbols are not known at the receiver:

- Constant timing offset, *i.e.*, $\tau_k = \tau_0 \; \forall \; k$.
- Frequency offset, *i.e.*, $\tau_k = \tau_0 + k\Delta T$.

- Random walk, *i.e.*, $\tau_k = \tau_{k-1} + w_k$ where $\{w_k\}$ are *i.i.d.* zero-mean normal random variables with variance σ_w^2.

- A combination of the three, *i.e.*, $\tau_k = \tau_{k-1} + \Delta T + w_k$ and τ_0 not necessarily zero.

Before we evaluate the CRB for these models, we make the following simplifying observation. Consider the system model of (50). The parameter $\boldsymbol{\theta}$ consists of two sets of parameters, the *data* parameters $\{a_k\}$ and the *timing* parameters $\{\tau_k\}$.

$$\begin{aligned} \boldsymbol{\theta} &= [a_0\ a_1\ \ldots\ a_{N-1} \mid \tau_0\ \tau_1\ \ldots\ \tau_{N-1}]^T, \\ &= [\boldsymbol{\theta}_d^T \mid \boldsymbol{\theta}_t^T]^T, \end{aligned} \tag{51}$$

where we have defined the data parameter vector $\boldsymbol{\theta}_d$ and the timing parameter vector $\boldsymbol{\theta}_t$. We assume that the noise n_k is *i.i.d.* normal and, therefore, the Fisher information matrix has the structure described in (48), where the mean $\boldsymbol{x}(\boldsymbol{\theta})$ of (45) is given by

$$[\boldsymbol{x}(\boldsymbol{\theta})]_k = \sum_{l=0}^{N-1} a_l h(kT - lT - \tau_l). \tag{52}$$

The matrix $\boldsymbol{G}$ can also be partitioned into data and timing portions, *i.e.*, $\boldsymbol{G} = [\boldsymbol{G}_d\ \boldsymbol{G}_t]$ and $\boldsymbol{J}_\theta$ has a block structure given by

$$\boldsymbol{J}_\theta = \frac{1}{\sigma^2} \begin{bmatrix} \boldsymbol{G}_d^T \boldsymbol{G}_d & \boldsymbol{G}_d^T \boldsymbol{G}_t \\ \boldsymbol{G}_t^T \boldsymbol{G}_d & \boldsymbol{G}_t^T \boldsymbol{G}_t \end{bmatrix}. \tag{53}$$

In the computation of the total information matrix, we need to take the expectation of $\boldsymbol{J}_\theta$ with respect to $\boldsymbol{\theta}$ (43). For the case where we know only the *a priori* distribution of the data symbols and not that of the timing parameters, we can still compute the total information matrix by making the timing component of $\boldsymbol{J}_a$ zero and assuming a uniform distribution on the timing parameters when we take the expectation of $\boldsymbol{J}_\theta$ with respect to $\boldsymbol{\theta}$. This is valid because not having an *a priori* distribution on the timing parameters is equivalent to having a uniform distribution and using this in (44)

leads to the timing components of $\boldsymbol{J}_a$ being zero. The total information matrix in this case is essentially the Fisher information averaged over all possible data sequences.

For our model, the data symbols are uniformly chosen from $\{\pm 1\}$ and this gives the *a priori* data distribution. Therefore, whether or not we have an *a priori* distribution on the timing parameters, we need to take the expectation of $\boldsymbol{J}_\theta$ with respect to $\boldsymbol{\theta}_d$ to arrive at the Cramér-Rao lower bound on the average estimation error variance of any unbiased timing estimator. Hence, we analyze $E[\boldsymbol{J}_\theta]$. Specifically, we study the behavior of the off-diagonal terms $E[\boldsymbol{G}_d^T \boldsymbol{G}_t]$ and $E[\boldsymbol{G}_t^T \boldsymbol{G}_d]$ of (53).

First, we construct the matrix $\boldsymbol{G}$, which is an $(N + 2M) \times P$ matrix. The data portion $\boldsymbol{G}_d$ has components:

$$[\boldsymbol{G}_d]_{ki} = \frac{\partial x_k}{\partial a_i} = h(kT - iT - \tau_i), \tag{54}$$

whereas the timing portion $\boldsymbol{G}_t$ has components:

$$\begin{aligned} [\boldsymbol{G}_t]_{ki} &= \frac{\partial x_k}{\partial \tau_i} \\ &= \sum_{l=0}^{N-1} a_l \frac{\partial}{\partial \tau_i} h(kT - lT - \tau_l) \\ &= -\sum_{l \in M_{ki}} a_l h'(kT - lT - \tau_l), \end{aligned} \tag{55}$$

where $h'(t)$ is the derivative of the pulse shape $h(t)$ and the index set M_{ki} depends on the actual timing model.

Combining (54) and (55), the terms of $E[\boldsymbol{G}_d^T \boldsymbol{G}_t]$ have the following general structure:

$$\begin{aligned} [\boldsymbol{G}_d^T \boldsymbol{G}_t]_{ij} &= \sum_{k=-M}^{N+M-1} [\boldsymbol{G}_d]_{ik} [\boldsymbol{G}_t]_{jk}, \\ &= -\sum_{k=-M}^{N+M-1} \left\{ \sum_{m \in M_{kj}} a_m h'(kT - mT - \tau_m) \right\} h(kT - iT - \tau_i). \end{aligned} \tag{56}$$

$E[\boldsymbol{G}_t^T \boldsymbol{G}_d]$ has a similar structure. Since the data symbols $\{a_k\}$ are zero-mean, taking expectation with respect to $\boldsymbol{\theta}_d$, we get

$$E[\boldsymbol{G}_d^T \boldsymbol{G}_t] = \boldsymbol{0}$$

$$E[\boldsymbol{G}_t^T \boldsymbol{G}_d] \;=\; \mathbf{0}. \tag{57}$$

Therefore, in the block-diagonal matrix $E[\boldsymbol{J}_\theta]$, the off-diagonal entries are zero. The inverse of this has a block diagonal structure with the diagonal entries being the inverses of the corresponding block diagonal entries of the original matrix, *i.e.*,

$$\begin{aligned} E[\boldsymbol{J}_\theta] &= \begin{bmatrix} \boldsymbol{J}_d & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J}_t \end{bmatrix} \quad \text{and} \\ E[\boldsymbol{J}_\theta]^{-1} &= \begin{bmatrix} \boldsymbol{J}_d^{-1} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{J}_t^{-1} \end{bmatrix}. \end{aligned} \tag{58}$$

If we are concerned only with the CRB on the timing parameters, then we only need to compute the block-diagonal entry of $E[\boldsymbol{J}_\theta]$ corresponding to the timing parameters and invert it. This is what we do in the following subsections for different timing models.

3.3.1 Constant Offset

We begin by calculating the CRB for constant offset, also done in [39]. With a constant offset, the uniform samples $\{r_k\}$ are given by

$$r_k = \sum_{l=0}^{N-1} a_l h(kT - lT - \tau) + n_k, \tag{59}$$

where τ is assumed to be a deterministic but unknown timing offset. In this case, $\boldsymbol{G}_t$ is a $(N + 2M) \times 1$ vector and is given by

$$[\boldsymbol{G}_t]_k = \frac{\partial x_k}{\partial \tau} = -\sum_{l=0}^{N-1} a_l h'(kT - lT - \tau). \tag{60}$$

Therefore, $\boldsymbol{J}_t$ is a scalar given by

$$J_t = \frac{1}{\sigma^2} \sum_{k=-M}^{N+M-1} \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} a_l a_m h'(kT - lT - \tau) h'(kT - mT - \tau). \tag{61}$$

Since the data symbols $\{a_k\}$ are *i.i.d.* and equally likely to be $\{\pm 1\}$, we get

$$E[J_t] = \frac{1}{\sigma^2} \sum_{l=0}^{N-1} \left\{ \sum_{k=-M}^{N+M-1} h'^2(kT - lT - \tau) \right\}. \tag{62}$$

Letting $M \to \infty$, we see that, because the sampling rate avoids aliasing, the discrete time energy equals the continuous time energy and, therefore, the bracketed summation is the energy in the derivative of the pulse shape $h(t)$. Denoting this by $E_{h'}$,

$$E[J_t] = \frac{NE_{h'}}{\sigma^2}, \tag{63}$$

and finally, we invert this to get the CRB which is [39]

$$\sigma_\epsilon^2 \geq \frac{\sigma^2}{NE_{h'}}, \tag{64}$$

where σ_ϵ^2 is the variance of the timing error $\epsilon = \tau - \hat{\tau}$, where $\hat{\tau}$ is an estimate of the timing offset τ. The CRB is inversely proportional to the number of samples N and directly proportional to the channel noise variance σ^2. The ratio σ^2/N can be viewed as the variance of the averaged noise, and the CRB is the averaged noise variance scaled down by the energy in the derivative of the channel response.

Example: Consider the system where the pulse shape is a 0% excess bandwidth pulse, *i.e.*,

$$h(t) = \frac{\sin(\pi \frac{t}{T})}{\pi \frac{t}{T}}. \tag{65}$$

To evaluate the CRB, we need the energy in $h'(t)$, the first derivative of $h(t)$, which is given by

$$h'(t) = \frac{\cos\left(\pi \frac{t}{T}\right) - \operatorname{sinc}\left(\frac{t}{T}\right)}{t}. \tag{66}$$

Using Fourier transform and Parseval's theorem, the energy in $h'(t)$ is

$$E_{h'} = \frac{\pi^2}{3T^2}. \tag{67}$$

Using this in the CRB expression of (64), we get

$$\frac{\sigma_\epsilon^2}{T^2} \geq \frac{3\sigma^2}{\pi^2 N}. \tag{68}$$

Similarly, for the PR-IV pulse given by

$$h(t) = \text{sinc}\left(\frac{t}{T}\right) - \text{sinc}\left(\frac{t-2T}{T}\right), \tag{69}$$

the CRB evaluates to

$$\frac{\sigma_\epsilon^2}{T^2} \geq \frac{\sigma^2}{\left(\frac{2\pi^2}{3} - 1\right) N}. \tag{70}$$

Next, assume that we have *a priori* information that $\tau \sim \mathcal{N}(0, \sigma_\tau^2)$. This additional information can be used to reduce the estimation error variance. The *a priori* distribution of the parameter τ is

$$f_\tau(\tau) = \frac{1}{\sqrt{2\pi}\sigma_\tau} \exp\left\{-\frac{\tau^2}{2\sigma_\tau^2}\right\}. \tag{71}$$

Therefore, the *a priori* information J_a is

$$J_a = E\left[\left(\frac{\partial}{\partial\tau} \ln f_\tau(\tau)\right)^2\right] = \frac{1}{\sigma_\tau^2}. \tag{72}$$

The total information is got by adding the average Fisher information of (63) and the *a priori* information of (72).

$$J_T = \frac{NE_{h'}}{\sigma^2} + \frac{1}{\sigma_\tau^2}. \tag{73}$$

Finally, we get the CRB by inverting J_T. Denote the CRB without any *a priori* information by CRB_{old} and the CRB with *a priori* information by CRB_{new}. Define the information factor f_τ as the ratio of the *a priori* information and the average Fisher information. That is, $f_\tau = \text{CRB}_{\text{old}}/\sigma_\tau^2$. In terms of these parameters,

$$\text{CRB}_{\text{new}} = \text{CRB}_{\text{old}} \frac{1}{1 + f_\tau}. \tag{74}$$

As the *a priori* information increases, that is, as σ_τ^2 decreases, f_τ increases and CRB_{new} goes to zero. The key observation is that the ratio $\frac{\text{CRB}_{\text{new}}}{\text{CRB}_{\text{old}}}$ is a function of f_τ. The form of $\frac{\text{CRB}_{\text{new}}}{\text{CRB}_{\text{old}}}$ conforms with the fact that the highest CRB is when we have no *a priori* information, corresponding to $f_\tau = 0$.

Example: Consider the system where the pulse shape is a 0% excess bandwidth pulse, *i.e.*,

$$h(t) = \frac{\sin(\pi \frac{t}{T})}{\pi \frac{t}{T}}. \tag{75}$$

In the previous example, the CRB without *a priori* information was evaluated to be

$$\text{CRB}_{\text{old}} = \frac{\sigma^2}{\left(\frac{2\pi^2}{3} - 1\right) N}. \tag{76}$$

Now we consider the utility of *a priori* information with a numerical example. Let the channel SNR be -3 dB, leading a channel noise variance of $\sigma^2 = 1$. Therefore CRB_{old} in this case is

$$\text{CRB}_{\text{old}} = \frac{0.1792}{N}. \tag{77}$$

First consider the case with only one observation, *i.e.*, $N = 1$. With *a priori* information $\frac{\sigma_\tau}{T} = 0.1$, we get $f_\tau = 1.792$, leading to $\text{CRB}_{\text{new}} = 0.358\text{CRB}_{\text{old}}$. On the other hand, with $\frac{\sigma_\tau}{T} = 0.01$, we get $f_\tau = 17.92$, leading to $\text{CRB}_{\text{new}} = 0.053\text{CRB}_{\text{old}}$. Therefore, the better the *a priori* information, the lower the CRB.

As N increases, the influence of the *a priori* information on the CRB reduces. To illustrate this, consider $N = 100$ for the system above, leading to $\text{CRB}_{\text{old}} = 1.792 \times 10^{-3}$. With $\frac{\sigma_\tau}{T} = 0.1$, we get $f_\tau = 1.792 \times 10^{-2}$, leading to $\text{CRB}_{\text{new}} = 0.982\text{CRB}_{\text{old}}$. On the other hand, with $\frac{\sigma_\tau}{T} = 0.01$, we get $f_\tau = 0.1792$, leading to $\text{CRB}_{\text{new}} = 0.848\text{CRB}_{\text{old}}$. The reduction in the CRB due to better *a priori* information is much lower when N is higher.

To illustrate this further, we can rewrite (74) as

$$\frac{1}{\text{CRB}_{\text{new}}} = \frac{1}{\text{CRB}_{\text{old}}} + \frac{1}{\sigma_\tau^2}. \tag{78}$$

Therefore, CRB_{new} is the harmonic mean of CRB_{old} and σ_τ^2. This is reminiscent of parallel addition of resistors, with CRB_{new} being the total resistance of a parallel combination of two resistors of values CRB_{old} and σ_τ^2 respectively. It is immediately obvious that a finite σ_τ^2 ensures that $\text{CRB}_{\text{new}} < \text{CRB}_{\text{old}}$. Also, as N increases, CRB_{old} reduces and, therefore, the effect of adding σ_τ^2 in parallel reduces.

3.3.2 Frequency Offset

Next, we consider the case where we have a frequency offset. This is equivalent to having a mismatch between the actual symbol duration T and the receiver's estimate T'. Let this difference be ΔT. First we assume that we have no *a priori* information about ΔT. Assuming that there is no initial timing offset, *i.e.*, $\tau_0 = 0$, the channel model is

$$r_k = \sum_{l=0}^{N-1} a_l h(kT - lT - k\Delta T) + n_k. \tag{79}$$

Proceeding as with the constant offset case, we get

$$E[J_t] = \frac{E_{h'}}{\sigma^2} \frac{(N-1)N(2N-1)}{6}, \tag{80}$$

leading to the following CRB:

$$\sigma_\epsilon^2 \geq \frac{6\sigma^2}{E_{h'}(N-1)N(2N-1)}. \tag{81}$$

As with the constant offset case, the CRB is inversely proportional to $E_{h'}$. As opposed to the constant offset case where the minimum estimation error variance was proportional to N^{-1}, with a frequency offset, the minimum estimation variance is asymptotically proportional to N^{-3}. With *a priori* information that $\Delta T \sim \mathcal{N}(0, \sigma_{\Delta T}^2)$, the new CRB is

$$\text{CRB}_{\text{new}} = \text{CRB}_{\text{old}} \frac{1}{1 + f_{\Delta T}}, \tag{82}$$

where $f_{\Delta T}$ is the ratio of the *a priori* information and the average Fisher information, *i.e.*, $f_{\Delta T} = \text{CRB}_{\text{old}}/\sigma_{\Delta T}^2$. This is the same form that we had in the constant offset

case.

Next, we assume that we have an initial timing offset τ_0 and a frequency offset parameter ΔT. The timing offsets $\{\tau_k\}$ are given by

$$\tau_k = \tau_{k-1} + \Delta T = \tau_0 + k\Delta T, \tag{83}$$

and the channel model is

$$r_k = \sum_{l=0}^{N-1} a_l h(kT - lT - \tau_0 - k\Delta T) + n_k. \tag{84}$$

We now have two parameters to be estimated. Let the timing parameter be $\boldsymbol{\theta}_t = [\Delta T \ \tau_0]$. Evaluating the average Fisher information, we get

$$E[\boldsymbol{J}_t] = \frac{E_{h'}}{\sigma^2} \begin{bmatrix} \frac{(N-1)N(2N-1)}{6} & \frac{N(N-1)}{2} \\ \frac{N(N-1)}{2} & N \end{bmatrix}. \tag{85}$$

Inverting this, the CRB is given by

$$E[\boldsymbol{J}_t]^{-1} = \frac{\sigma^2}{E_{h'}} \begin{bmatrix} \frac{12}{(N-1)N(N+1)} & \frac{-6}{N(N+1)} \\ \frac{-6}{N(N+1)} & \frac{2(2N-1)}{N(N+1)} \end{bmatrix}. \tag{86}$$

The usual parameters of interest are the estimation error variance of the frequency offset parameter ΔT and also that of the initial offset τ_0, and these are the diagonal elements of $\boldsymbol{J}_t^{-1}$. Writing it out, we get

$$\begin{aligned} E[(\Delta T - \hat{\Delta T})^2] &\geq \frac{12\sigma^2}{E_{h'}(N-1)N(N+1)} \\ E[(\tau_0 - \hat{\tau_0})^2] &\geq \frac{2\sigma^2(2N-1)}{E_{h'}N(N+1)}. \end{aligned} \tag{87}$$

The error variance for ΔT goes asymptotically as N^{-3} whereas the error variance for τ_0 goes as N^{-1}. It is interesting to compare the CRB of τ_0 in this case, $\mathrm{CRB}_{i+f}^{\tau_0}$, where the subscript $i+f$ stands for the fact that we have an initial timing offset and a frequency offset, with the error variance when we had a constant offset, $\mathrm{CRB}_{i}^{\tau_0}$, where the subscript i denotes the presence of only the initial timing offset. This ratio ζ turns out to be

$$\zeta = \frac{\mathrm{CRB}_{i+f}^{\tau_0}}{\mathrm{CRB}_{i}^{\tau_0}} = \frac{2(2N-1)}{N+1}. \tag{88}$$

When $N = 1$, we have $\zeta = 1$. ζ is a monotonically increasing function of N and as N becomes large, this tends to $\zeta = 4$. In other words, while estimating the initial timing offset, we face an asymptotic penalty factor of 4 due to the presence of the frequency offset. The penalty factor for estimating ΔT is the same as ζ, *i.e.*,

$$\frac{\mathrm{CRB}_{i+f}^{\tau_0}}{\mathrm{CRB}_{i}^{\tau_0}} = \frac{\mathrm{CRB}_{i+f}^{\Delta T}}{\mathrm{CRB}_{f}^{\Delta T}} = \zeta = \frac{2(2N-1)}{N+1}. \tag{89}$$

Next we evaluate the value of *a priori* information regarding the parameters to be estimated. Assume we have *a priori* knowledge that $\Delta T \sim \mathcal{N}(0, \sigma_{\Delta T}^2)$ and $\tau_0 \sim \mathcal{N}(0, \sigma_{\tau_0}^2)$. The total information matrix is given by

$$\boldsymbol{J}_T = \begin{bmatrix} \frac{E_{h'}}{\sigma^2}\frac{(N-1)N(2N-1)}{6} + \frac{1}{\sigma_{\Delta T}^2} & \frac{E_{h'}}{\sigma^2}\frac{N(N-1)}{2} \\ \frac{E_{h'}}{\sigma^2}\frac{N(N-1)}{2} & \frac{E_{h'}}{\sigma^2}N + \frac{1}{\sigma_{\tau_0}^2} \end{bmatrix}. \tag{90}$$

Inverting $\boldsymbol{J}_T$, we get the CRB. Denote the CRB for estimating τ_0 without any *a priori* knowledge by $\mathrm{CRB}_{\mathrm{old}}^{\tau_0}$. (This is what was called $\mathrm{CRB}_{i+f}^{\tau_0}$ earlier.) Similarly define $\mathrm{CRB}_{\mathrm{old}}^{\Delta T}$. Next, define parameters $f_{\Delta T} = \mathrm{CRB}_{\mathrm{old}}^{\Delta T}/\sigma_{\Delta T}^2$ and $f_{\tau_0} = \mathrm{CRB}_{\mathrm{old}}^{\tau_0}/\sigma_{\tau_0}^2$ which are a measure of *a priori* information relative to the average Fisher information. In terms of these parameters and ζ, the new CRB (including the *a priori* information) evaluates to

$$\begin{aligned} \mathrm{CRB}_{\mathrm{new}}^{\Delta T} &= \mathrm{CRB}_{\mathrm{old}}^{\Delta T}\,\frac{1+\frac{f_{\tau_0}}{\zeta}}{1+f_{\Delta T}+f_{\tau_0}+\frac{f_{\Delta T}f_{\tau_0}}{\zeta}} \quad \text{and} \\ \mathrm{CRB}_{\mathrm{new}}^{\tau_0} &= \mathrm{CRB}_{\mathrm{old}}^{\tau_0}\,\frac{1+\frac{f_{\Delta T}}{\zeta}}{1+f_{\Delta T}+f_{\tau_0}+\frac{f_{\Delta T}f_{\tau_0}}{\zeta}}. \end{aligned} \tag{91}$$

This structure is a little more complicated than the parallel addition seen in the case of a constant offset. To gain insight into these expressions, consider the two extreme cases of *a priori* information given by $f_{\tau_0} = 0$ (no *a priori* information) and $f_{\tau_0} = \infty$ (complete *a priori* information).

$$\begin{aligned} f_{\tau_0} = 0 \qquad &\Longrightarrow \\ \mathrm{CRB}_{\mathrm{new}}^{\Delta T} &= \mathrm{CRB}_{\mathrm{old}}^{\Delta T}\,\frac{1}{1+f_{\Delta T}} \end{aligned} \tag{92}$$

$$\mathrm{CRB}^{\tau_0}_{\mathrm{new}} = \mathrm{CRB}^{\tau_0}_{\mathrm{old}} \frac{1 + \frac{f_{\Delta T}}{\zeta}}{1 + f_{\Delta T}} \tag{93}$$

$$f_{\tau_0} = \infty \quad \Longrightarrow$$

$$\mathrm{CRB}^{\Delta T}_{\mathrm{new}} = \mathrm{CRB}^{\Delta T}_{\mathrm{old}} \frac{1}{\zeta + f_{\Delta T}} \tag{94}$$

$$\mathrm{CRB}^{\tau_0}_{\mathrm{new}} = 0 \tag{95}$$

From (92), we see that when we have no *a priori* information about τ_0, the gain due to *a priori* information about ΔT is the same as in (82) where we had only a frequency offset, and no initial timing offset. Comparing (92) and (94) for a fixed $f_{\Delta T}$, $\mathrm{CRB}^{\Delta T}_{\mathrm{new}}|_{f_{\tau_0}=\infty} < \mathrm{CRB}^{\Delta T}_{\mathrm{new}}|_{f_{\tau_0}=0}$, *i.e.*, *a priori* information about τ_0 helps in estimating ΔT. An alternative conclusion that can be drawn from this is that when we have two parameters that are coupled through the same observations, it helps to exploit this coupling rather than estimate the parameters separately. The same conclusion can be drawn from (93) where a non-zero $f_{\Delta T}$ makes $\mathrm{CRB}^{\tau_0}_{\mathrm{new}} < \mathrm{CRB}^{\tau_0}_{\mathrm{old}}$ even when $f_{\tau_0} = 0$. Finally, (95) states the obvious fact that when we have complete *a priori* information, the estimation error variance is lower bounded by zero.

3.3.3 Accumulation Process

Next, we look at the accumulation process where the timing offsets are as follows:

$$\tau_{k+1} = \tau_k + w_k = \sum_{j=0}^{k} w_j, \tag{96}$$

where we assume $\tau_0 = 0$ and the sequence $\{\tau_k\}$ is formed by accumulating $\{w_k\}$. If we assume that the elements of $\{w_k\}$ are *i.i.d.* zero-mean normal random variables, we get the random walk model which we study in the next section. For the present, we assume that we have no *a priori* information about the variables $\{w_k\}$. The channel model with uniform sampling is

$$\begin{aligned} r_k &= \sum_{l=0}^{N-1} a_l h(kT - lT - \tau_l) + n_k \\ &= \sum_{l=0}^{N-1} a_l h(kT - lT - \sum_{j=0}^{l-1} w_j) + n_k \end{aligned}$$

$$= \quad x_k + n_k, \tag{97}$$

where x_k is again the noiseless received value. The parameter to be estimated is $\boldsymbol{\tau} = [\tau_1 \ \tau_2 \ \ldots \ \tau_{N-1}]^T$. Instead of directly computing the Fisher information matrix $\boldsymbol{J}_\tau$, we define the parameter $\boldsymbol{w} = [w_0 \ w_1 \ \ldots \ w_{N-2}]^T$, compute $\boldsymbol{J}_w$, and then use the linear transformation relating $\boldsymbol{\tau}$ and $\boldsymbol{w}$ to get $\boldsymbol{J}_\tau$.

We proceed with evaluating the matrix $\boldsymbol{G}$ for the parameter $\boldsymbol{w}$. We have

$$g_{ki} = \frac{\partial x_k}{\partial w_i} = \sum_{l=0}^{N-1} a_l \frac{\partial}{\partial w_i} h(kT - lT - \sum_{j=0}^{l-1} w_j). \tag{98}$$

The derivative of $h(kT - lT - \sum_{j=0}^{l-1} w_j)$ with respect to w_i is zero if w_i does not occur in the summation inside the $h(\cdot)$ term. Therefore,

$$g_{ki} = -\sum_{l=i+1}^{N-1} a_l h'(kT - lT - \sum_{j=0}^{l-1} w_j) = -\sum_{l=i+1}^{N-1} a_l h'(kT - lT - \tau_l). \tag{99}$$

Next, we compute $\boldsymbol{G}^T\boldsymbol{G}$:

$$\begin{aligned} [\boldsymbol{G}^T\boldsymbol{G}]_{i_1 i_2} &= \sum_{k=-M}^{N+M-1} g_{ki_1} g_{ki_2} \\ &= \sum_{k=-M}^{N+M-1} \sum_{j_1=i_1+1}^{N-1} \sum_{j_2=i_2+1}^{N-1} a_{j_1} a_{j_2} h'(kT - j_1 T - \tau_{j_1}) h'(kT - j_2 T - \tau_{j_2}) \end{aligned} \tag{100}$$

To get the averaged Fisher information, we need to take the expectation of $\boldsymbol{G}^T\boldsymbol{G}$ over all data sequences $\boldsymbol{a}$. Since the data symbols are assumed to be uncorrelated, we have $E[a_{j_1} a_{j_2}] = \delta_{j_1 j_2}$. Using this fact, the second and third sums in (100) collapse to a single summation, and, changing the order of summation,we get

$$E[\boldsymbol{G}^T\boldsymbol{G}]_{i_1 i_2} = \sum_{j=\max(i_1,i_2)+1}^{N-1} \left\{ \sum_{k=-M}^{N+M-1} [h'(kT - jT - \tau_j)]^2 \right\}, \tag{101}$$

where the expectation is over $\boldsymbol{a}$. Now ,letting $M \to \infty$ we see that the bracketed term is the energy in $h'(t)$, denoted by $E_{h'}$. Therefore,

$$E[\boldsymbol{J}_w]_{i_1 i_2} = \frac{E_{h'}}{\sigma^2} \left((N-1) - \max(i_1, i_2) \right). \tag{102}$$

In long hand,

$$E[\boldsymbol{J}_w] = \frac{E_{h'}}{\sigma^2} \begin{bmatrix} N-1 & N-2 & N-3 & \dots & 1 \\ N-2 & N-2 & N-3 & \dots & 1 \\ N-3 & N-3 & N-3 & \dots & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix}. \tag{103}$$

To get $\boldsymbol{J}_\tau$ from $\boldsymbol{J}_w$, we use the relationship between $\boldsymbol{\tau}$ and $\boldsymbol{w}$. Recall that $\tau_k = \sum_{j=0}^{k-1} w_j$. In vector notation,

$$\boldsymbol{\tau} = \boldsymbol{T}\boldsymbol{w}, \tag{104}$$

where

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ 1 & 1 & 0 & \dots & 0 \\ 1 & 1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \ddots & 0 \\ 1 & 1 & 1 & \dots & 1 \end{bmatrix} \quad \text{is a lower triangular matrix.} \tag{105}$$

Using this relationship between $\boldsymbol{\tau}$ and $\boldsymbol{w}$, we can write [58]

$$\boldsymbol{J}_\tau = \boldsymbol{T}^{-T}\boldsymbol{J}_w\boldsymbol{T}^{-1}. \tag{106}$$

To compute $\boldsymbol{T}^{-1}$, recognize the fact that we need now the inverse mapping of $\tau_k = \sum_{j=0}^{k-1} w_j$, which is $w_k = \tau_{k+1} - \tau_k$. Therefore,

$$\boldsymbol{T}^{-1} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & -1 & 1 \end{bmatrix}, \tag{107}$$

which is a lower triangular Toeplitz matrix with only two non-zero diagonals, the main diagonal and an adjacent one. Using (103), (106) and (107) we get

$$E[\boldsymbol{J}_\tau] = \frac{E_{h'}}{\sigma^2}\boldsymbol{I}_{N\times N}. \tag{108}$$

Inverting this to get the CRB is simple and we get

$$E[(\tau - \hat{\tau})(\tau - \hat{\tau})^T] \geq \frac{\sigma^2}{E_{h'}} \boldsymbol{I}_{N \times N}. \tag{109}$$

This result implies that for an accumulation process with no *a priori* information, the bound on the estimation error for each component is the same and equal to $\sigma^2/E_{h'}$, which is the same as the CRB for estimating a constant offset with $N = 1$, *i.e.*, the one-shot transmission case. For the accumulation process, the CRB is independent of N. This result makes intuitive sense since each new observation brings along with it a new parameter to be estimated.

3.3.4 Random Walk

When we put a specific *a priori* distribution on the accumulation process, we get the random walk. We consider the random walk given by

$$\tau_{k+1} = \tau_k + w_k = \tau_0 + \sum_{j=0}^{k} w_j, \tag{110}$$

where $w_k \sim \mathcal{N}(0, \sigma_w^2)$ are *i.i.d.* and σ_w^2 determines the severity of the timing jitter. We assume perfect acquisition, *i.e.*, $\tau_0 = 0$.

We now need to compute the total information, which is the sum of the average Fisher information of (108) and the *a priori* information. The *a priori* distribution $\boldsymbol{f}_{\boldsymbol{w}}(\boldsymbol{w})$ is

$$f_{\boldsymbol{w}}(\boldsymbol{w}) = \prod_{j=0}^{N-2} \frac{1}{\sqrt{2\pi\sigma_w^2}} \exp\left(-\frac{w_j^2}{2\sigma_w^2}\right). \tag{111}$$

Therefore, the *a priori* information matrix for $\boldsymbol{w}$ is

$$\boldsymbol{J}_a{}_{\boldsymbol{w}} = \frac{1}{\sigma_w^2} \boldsymbol{I}, \tag{112}$$

and the *a priori* information matrix for $\boldsymbol{\tau}$ is [58]

$$\boldsymbol{J}_a{}_{\boldsymbol{\tau}} = \frac{1}{\sigma_w^2} \boldsymbol{T}^{-T} \boldsymbol{T}^{-1}, \tag{113}$$

where $\boldsymbol{T}$ is the linear transformation relating $\boldsymbol{w}$ and $\boldsymbol{\tau}$ defined in (105). Combining (43), (108) and (113) and simplifying, we get

$$\boldsymbol{J}_T = \frac{1}{\sigma_w^2}\begin{bmatrix} \lambda & -1 & 0 & \dots & 0 \\ -1 & \lambda & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & \lambda & -1 \\ 0 & \dots & 0 & -1 & \lambda - 1 \end{bmatrix}, \text{ where } \lambda = 2 + E_{h'}\frac{\sigma_w^2}{\sigma^2}. \tag{114}$$

We need to invert $\boldsymbol{J}_T$ to get the CRB. The details of the inversion procedure are given in Appendix B. To get the CRB on the estimation error for the individual timing estimates τ_k, we need the diagonal elements of $[\boldsymbol{J}_T]^{-1}$. The CRB for the random walk evaluates to

$$\frac{E[(\hat{\tau}_i(\boldsymbol{r}) - \tau_i)^2]}{T^2} \geq \frac{[\boldsymbol{J}_T]^{-1}_{ii}}{T^2} = h \cdot f(i), \tag{115}$$

where

$$\begin{aligned} h &= \frac{\sigma_w^2}{T^2}\frac{\eta}{\eta^2 - 1} \quad \text{is the steady state value,} \\ f(i) &= \tanh\left((N + \frac{1}{2})\ln\eta\right)\left[1 - \frac{\sinh\left((N - 2i + \frac{1}{2})\ln\eta\right)}{\sinh\left((N + \frac{1}{2})\ln\eta\right)}\right], \\ \eta &= \frac{\lambda + \sqrt{\lambda^2 - 4}}{2} \quad \text{and} \quad \lambda = 2 + E_{h'}\frac{\sigma_w^2}{\sigma^2}. \end{aligned} \tag{116}$$

Example: Consider again the PR-IV system with

$$h(t) = \text{sinc}\left(\frac{t}{T}\right) - \text{sinc}\left(\frac{t - 2T}{T}\right), \tag{117}$$

for which $E_{h'} = \frac{1}{T^2}\left(\frac{2\pi^2}{3} - 1\right)$.

Figure 22 plots the bound as a function of the symbol index k for SNR = 5 dB and for $\sigma_w/T \in \{0.005\%,\ 0.05\%,\ 0.5\%,\ 5\%\}$. Along with the bounds, we have plotted a horizontal line at value h. As σ_w/T increases, we see that the shaping function $f(i)$ tends towards a constant, independent of

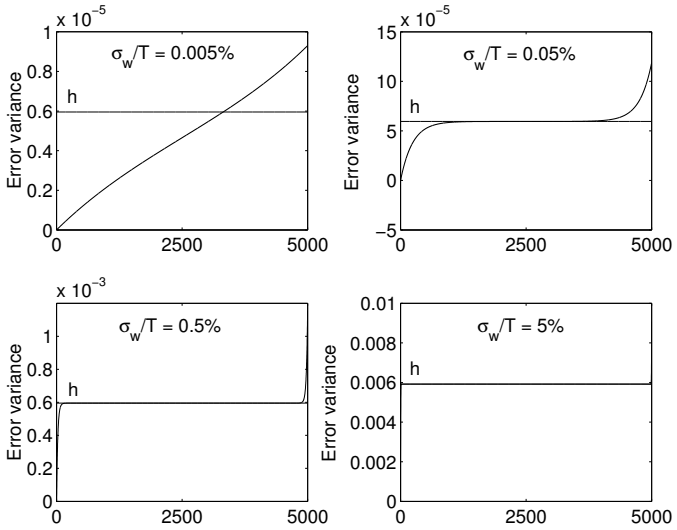

Figure 22: The lower bound on timing estimation error variance at SNR = 5.0 dB.

i. The steady state bound can be rewritten as

$$h = \frac{\sigma \frac{\sigma_w}{T}}{\sqrt{K_1(4 + K_1 \frac{\sigma_w^2}{\sigma^2 T^2})}}, \tag{118}$$

where $K_1 = (2\pi^2/3) - 1$. For $\sigma_w^2/T^2 \ll \sigma^2$, which would be the case, for example, with $SNR \sim 5dB$ and $\sigma_w/T < 10\%$, we can approximate

$$h \approx K_2 \frac{\sigma_w}{T} \sigma, \tag{119}$$

where $K_2 = 1/(2\sqrt{K_1})$. Therefore, the steady state lower bound is linear in both σ_w/T and σ. If we look at the other extreme $\sigma_w^2/T^2 \gg \sigma^2$, we get

$$h \approx \frac{\sigma^2}{K_1}, \tag{120}$$

independent of σ_w/T. This behavior is illustrated in Figure 23. When σ_w^2/T^2 is large, we have very little *a priori* information about the parameters to be estimated, and therefore, it is not surprising that the CRB approaches $\frac{\sigma^2}{K_1}$ which is the CRB associated with the accumulation process with no *a priori* information.

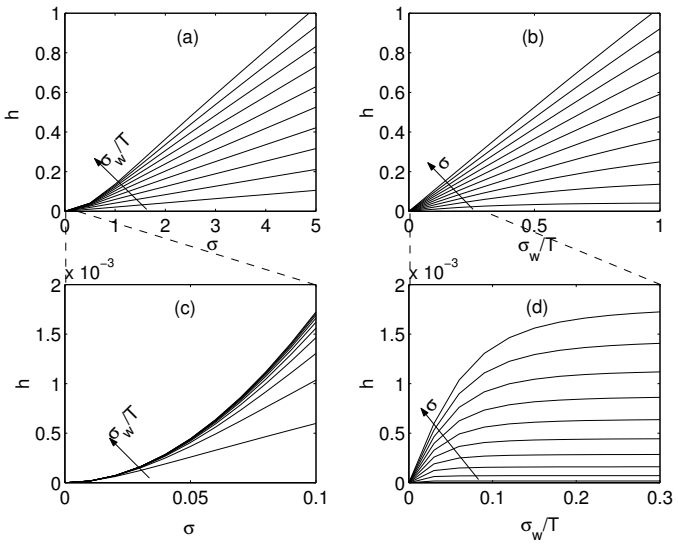

Figure 23: Steady state bound at the two extremes.

Figure 23-(a) plots h *vs.* σ for σ_w/T going from 0.1 to 1 in steps of 0.1. The arrow shows the direction of increasing σ_w/T. Figure 23-(b) plots h *vs.* σ_w/T for σ from 0.5 to 5 in steps of 0.5. This is the case where we have $\sigma_w^2/T^2 \ll \sigma^2$, and h is linear in both σ and σ_w/T as given by (119). Figures 23-(c) and 23-(d) deal with the other extreme. In Figure 23-(c), we have h *vs.* σ for σ_w/T going from 0.03 to 0.3 in steps of 0.03, and in Figure 23-(d), we have h *vs.* σ_w/T for σ from 0.01 to 0.1 in steps of 0.01. We see that whenever $\sigma_w^2/T^2 \ll \sigma^2$, h is quadratic in σ and independent of σ_w/T, as given by (120).

Consider next the shaping function $f(i)$. We can expect this to be a non-decreasing function of the symbol index i, due to the random walk model being used for the timing jitter. Since we start off with perfect acquisition, we expect the variance to rise from 0, as is indeed the case. As i increases, we reach the steady state bound. But as we approach the end of the packet, we see a curious exponential deviation from the steady

state value, governed by

$$\frac{[\boldsymbol{J}_1^{11}]^{-1}_{(N-j)(N-j)}}{T^2} - h \approx \frac{\sigma_w^2}{T^2}\frac{\eta^{2-2j}}{\eta^2-1}, \tag{121}$$

the shape of which is independent of N as shown in Figure 24, where SNR = 5 dB and $\sigma_w/T = 0.05\%$. As N increases, this end effect affects a smaller and smaller fraction of symbols. In other words, the steady state bound becomes more representative.

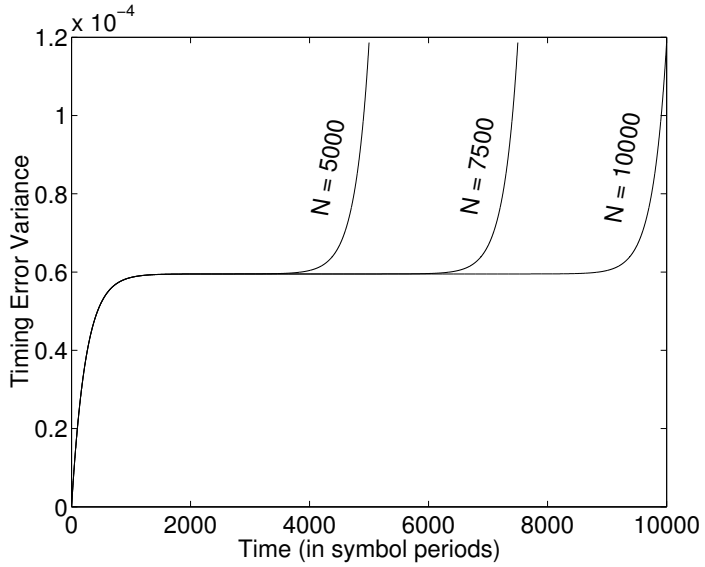

Figure 24: Shape of the end effect is independent of packet length.

Let r be the ratio of the variance of the last symbol to the steady state value. From (115) and (121), $r \approx 1 + \eta^{-1}$. From (116), $\eta \geq 1$. When $\sigma_w^2/T^2 \ll \sigma^2$, $\eta \approx 1$ and therefore, $r \approx 2$. When $\sigma_w^2/T^2 \gg \sigma^2$, $\eta \gg 1$ and therefore, $r \approx 1$. So, for any particular σ_w/T, as SNR ranges from $-\infty$ to ∞, r goes from 2 to 1. In other words, the end effect becomes less and less significant as SNR increases.

3.3.5 General Model: Frequency Offset + Random Walk

Next, we consider the general timing model given by

$$\tau_{k+1} = \tau_k + \Delta T + w_k = \tau_0 + (k+1)\Delta T + \sum_{j=0}^{K} w_j, \tag{122}$$

where τ_0 is the initial timing offset, ΔT is the frequency offset parameter and $\{w_k\}$ characterize the random walk where $\{w_k\}$ are *i.i.d.* zero-mean normal random variables of variance σ_w^2. The *a priori* distribution of τ_0 is $\mathcal{N}(0, \sigma_{\tau_0}^2)$ and that of ΔT is $\mathcal{N}(0, \sigma_{\Delta T}^2)$. To compute the average Fisher information, we recognize that this is an accumulation process and, therefore, from (108),

$$E[\boldsymbol{J}_\tau] = \frac{E_{h'}}{\sigma^2}\boldsymbol{I}_{N\times N}. \tag{123}$$

To get the *a priori* information matrix, we follow the same procedure as with the random walk model. We define a new parameter $\boldsymbol{\phi} = [\Delta T\ \tau_0\ w_0\ w_1\ \ldots\ w_{N-2}]^T$, get the *a priori* information matrix $\boldsymbol{J}_{a\boldsymbol{\phi}}$, and finally use the relationship between $\boldsymbol{\phi}$ and $\boldsymbol{\tau}$ to get $\boldsymbol{J}_{a\boldsymbol{\tau}}$.

The *a priori* distribution $\boldsymbol{f}_{\boldsymbol{\phi}}(\boldsymbol{\phi})$ is

$$f_{\boldsymbol{\phi}}(\boldsymbol{\phi}) = \frac{1}{\sqrt{2\pi\sigma_{\Delta T}^2}}\exp\left(-\frac{(\Delta T)^2}{2\sigma_{\Delta T}^2}\right)\frac{1}{\sqrt{2\pi\sigma_{\tau_0}^2}}\exp\left(-\frac{\tau_0^2}{2\sigma_{\tau_0}^2}\right)\prod_{j=0}^{N-2}\frac{1}{\sqrt{2\pi\sigma_w^2}}\exp\left(-\frac{w_j^2}{2\sigma_w^2}\right). \tag{124}$$

Therefore, the *a priori* information matrix for $\boldsymbol{\phi}$ is

$$\boldsymbol{J}_{a\boldsymbol{\phi}} = \begin{bmatrix} \frac{1}{\sigma_{\Delta T}^2} & & & & \\ & \frac{1}{\sigma_{\tau_0}^2} & & & \\ & & \frac{1}{\sigma_w^2} & & \\ & & & \ddots & \\ & & & & \frac{1}{\sigma_w^2} \end{bmatrix}. \tag{125}$$

The parameter vectors $\boldsymbol{\tau}$ and $\boldsymbol{\phi}$ are related as $\boldsymbol{\tau} = \boldsymbol{M}\boldsymbol{\phi}$, where

$$\boldsymbol{M} = \begin{bmatrix} 0 & 1 & 0 & 0 & \cdots & 0 \\ 1 & 1 & 1 & 0 & \cdots & 0 \\ 2 & 1 & 1 & 1 & \ddots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \ddots & 0 \\ N-1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix}. \tag{126}$$

The *a priori* information matrices $\boldsymbol{J}_{a\boldsymbol{\tau}}$ and $\boldsymbol{J}_{a\boldsymbol{\phi}}$ are related by [63]

$$\boldsymbol{J}_{a\boldsymbol{\phi}} = \boldsymbol{M}^T \boldsymbol{J}_{a\boldsymbol{\tau}} \boldsymbol{M}. \tag{127}$$

Since $\boldsymbol{M}$ is not a square matrix, we need a pseudo-inverse of $\boldsymbol{M}$ to get $\boldsymbol{J}_{a\boldsymbol{\tau}}$ from $\boldsymbol{J}_{a\boldsymbol{\phi}}$. Let $\boldsymbol{M}^\dagger = \boldsymbol{M}^T(\boldsymbol{M}\boldsymbol{M}^T)^{-1}$ be the right pseudo-inverse of $\boldsymbol{M}$ such that $\boldsymbol{M}\boldsymbol{M}^\dagger = \boldsymbol{I}$. Then,

$$\boldsymbol{J}_{a\boldsymbol{\tau}} = \boldsymbol{M}^{\dagger T} \boldsymbol{J}_{a\boldsymbol{\phi}} \boldsymbol{M}^\dagger. \tag{128}$$

The pseudo-inverse $\boldsymbol{M}^\dagger$ evaluates to

$$\boldsymbol{M}^\dagger = \begin{bmatrix} -\frac{1}{N} & & & & \\ 1 & & & & \\ -1+\frac{1}{N} & 1 & & & \\ \frac{1}{N} & -1 & \ddots & & \\ \vdots & & \ddots & \ddots & \\ \frac{1}{N} & & & -1 & 1 \end{bmatrix}. \tag{129}$$

Using (125) and (129), we evaluate the *a priori* information matrix $\boldsymbol{J}_{a\boldsymbol{\tau}}$. Adding this to the average Fisher information matrix, we get

$$\boldsymbol{J}_T = \frac{1}{\sigma_w^2} \begin{bmatrix} \beta & -1 & 0 & \dots & 0 \\ -1 & \lambda & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & \lambda & -1 \\ 0 & \dots & 0 & -1 & \lambda - 1 \end{bmatrix}, \text{ where} \tag{130}$$

$$\begin{aligned}\lambda &= 2 + E_{h'}\frac{\sigma_w^2}{\sigma^2} \quad \text{and} \\ \beta &= \left(\frac{N^2 - N - 1}{N^2} + \frac{\sigma_w^2}{\sigma_{\tau_0}^2} + \frac{\sigma_w^2}{N^2\sigma_{\Delta T}^2}\right) + E_{h'}\frac{\sigma_w^2}{\sigma^2}. \end{aligned} \tag{131}$$

Observe that the *a priori* information parameters $\sigma_{\tau_0}^2$ and $\sigma_{\Delta T}^2$ occur in the denominator of the fractions in β. When these are zero, this means we have perfect knowledge of the initial timing offset and the frequency offset and the estimation problem is simply that of estimating the random walk. Therefore, we assume that $\sigma_{\tau_0}^2$ and $\sigma_{\Delta T}^2$ are non-zero with the understanding that the zero case can be handled by eliminating suitable parameters from the estimation problem. Proceeding as with the random walk case, we get the CRB to be

$$[\boldsymbol{J}_T]_{ij}^{-1} = \sigma_w^2 \frac{a_{\max(i,j)} n_{\min(i,j)}}{n_N - n_{N-1}}, \text{ where} \tag{132}$$

$$\begin{aligned} a_j &= \frac{\eta^{N-j} + \eta^{-N+1+j}}{\eta + 1}, \\ n_j &= \frac{(\beta\eta - 1)\eta^j + (\eta^2 - \beta\eta)\eta^{-j}}{\eta^2 - 1} \quad \text{and} \\ \eta &= \frac{\lambda + \sqrt{\lambda^2 - 4}}{2}. \end{aligned} \tag{133}$$

3.4 Summary

In this chapter, we derived the CRB on the timing estimation error variance for different timing models, namely the constant offset, frequency offset and the random walk cases. For the constant offset case, the CRB is proportional to N^{-1} and for the frequency offset case, the CRB is proportional to N^{-3}. For the accumulation process, the CRB is independent of N and for each element in the parameter array the CRB is the same as that of estimating a single timing parameter associated with the one-shot transmission case. This makes intuitive sense since, in an accumulation process, each new observation also brings with it a new parameter to be estimated.

The CRB is a bound on the estimation error variance of deterministic but unknown quantities. In practice, we usually have some *a priori* information about the

parameters to be estimated. We analyzed the effect of *a priori* information on these estimation problems and showed that *a priori* information can only reduce the CRB. The factor by which the CRB improves as a result of the *a priori* information is a function of the ratio of the *a priori* information to the CRB without any *a priori* information. To estimate multiple parameters based on the same set of observations, it is beneficial to exploit this coupling instead of estimating the two parameters independently.

For the random walk case, the CRB is an increasing function of the symbol index, and, for a broad range of system parameters, exhibits a steady-state behavior in that it is almost a constant except towards the beginning and the end of the packet. We call this constant the steady-state value. The steady-state value becomes more representative as the block length increases. With low SNR, the steady-state CRB is proportional to the channel noise standard deviation and also to the random walk input standard deviation. With a severe random walk, the steady-state CRB is independent of the random walk parameter and approaches the CRB of the accumulation process with no *a priori* information. Finally, we derived the CRB for a general timing model consisting of an initial timing offset, a frequency offset and a random walk.

CHAPTER 4

CONVENTIONAL TIMING RECOVERY

In this chapter, we review conventional timing recovery which is based on the phase-locked loop (PLL) [64]. The PLL generates sampling instants based on the timing error estimates generated by a timing error detector (TED). We consider two common kinds of PLL: the first-order PLL and the second-order PLL. As a specific instance of the TED, we consider the well-known Müller and Mueller (MM) [44] TED which uses two previous samples and the corresponding decisions to estimate the timing error.

We compare the conventional timing recovery method with the CRB derived in Chapter 3 and observe that the it does not achieve the CRB. In an effort to bridge the gap between the PLL-based method and the CRB, we study the Kalman filtering approach described in [53] and [21]. It has been shown that the Kalman filter for estimating a general timing offset model, involving an initial timing offset, a frequency offset and a random walk, takes the form of a PLL with time-varying gains ([53] and [21]). The Kalman filter is the optimal causal filter with Gaussian noise, in that it minimizes the mean square estimation error. We derive the CRB for timing estimation under the causality constraint, and show that the Kalman filter indeed achieves this.

In the tracking mode, it has been shown that the Kalman filter reduces to a PLL with constant gains [53] operating on the output of a TED. Therefore, once the optimal PLL gains are chosen, a limiting factor to the performance of the PLL-based architecture is the TED. Thus, one method to improve the performance of PLL-based timing recovery is to improve the TED performance. We do this by keeping the TED architecture the same but passing better decisions to it, as described in Chapter 7. Alternatively, we could improve performance by relaxing the causality constraint and

performing block processing. This is described in Chapter 5.

The PLL-based system is used in the decision-directed mode when the system has already locked on to the desired stable operating point. Locking on to the desired operating point is done during acquisition. A method for acquisition is to insert known symbols at the start of transmission and then use these at the receiver to perform correlation detection to get coarse timing estimates and then refine these estimates using the PLL in the trained mode. Hence, the study of the PLL applies to both the tracking and acquisition modes.

The rest of the chapter is organized as follows. The conventional PLL-based timing recovery method is reviewed in Section 4.1. In Section 4.2, we compare the performance of the conventional timing recovery method with the CRB. The equivalence between the PLL and the Kalman filter is discussed in Section 4.3, and finally, the chapter is summarized in Section 4.4.

4.1 PLL-based Timing Recovery

We consider the same system that was used to derive the CRB. The block diagram is shown in Figure 21 and is reproduced here in Figure 25.

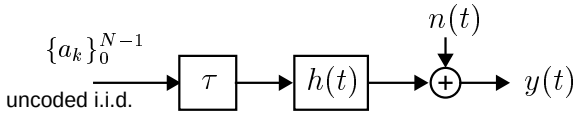

Figure 25: System block diagram with timing offsets, channel distortion and additive noise.

The channel output waveform $y(t)$ is given by

$$y(t) = \sum_{l=0}^{N-1} a_l h(t - lT - \tau_l) + n(t), \tag{134}$$

where T is the bit period, $a_l \in \{\pm 1\}$ are the N *i.i.d.* data symbols, $h(t)$ is the channel impulse response, $n(t)$ is additive white Gaussian noise, and τ_l is the unknown timing

offset for the l^{th} symbol. After low-pass filtering $y(t)$ at the receiver to remove the out-of-band noise, the resulting continuous-time waveform $r(t)$ can be modeled as

$$r(t) = \sum_l a_l h(t - lT - \tau_l) + n_1(t), \tag{135}$$

where $n_1(t)$ is band-limited to $[-\frac{1}{2T}, \frac{1}{2T})$. The continuous-time waveform $r(t)$ is then sampled at timing instants $kT + \hat{\tau}_k$ based on the estimates $\{\hat{\tau}_k\}$ of $\{\tau_k\}$ produced by the timing recovery system. Ideally we would like to sample at instants $kT + \tau_k$.

Conventional timing recovery is based on a PLL. A first-order PLL updates its estimate of τ_k according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha\hat{\epsilon}_k, \tag{136}$$

where α is the PLL gain, and where $\hat{\epsilon}_k$ is the receiver's estimate of the estimation error $\epsilon_k = \tau_k - \hat{\tau}_k$. For zero steady-state error while tracking a frequency offset, we employ a second-order PLL which updates the estimate of τ_k according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha\hat{\epsilon}_k + \beta\sum_{l=0}^{k-1}\hat{\epsilon}_l, \tag{137}$$

where we have an additional gain parameter β.

The receiver employs a timing error detector (TED) to arrive at timing error estimates. The widely-used Müller and Mueller (MM) TED [44] generates $\hat{\epsilon}_k$ according to

$$\hat{\epsilon}_k = \frac{3T}{16} r_k \hat{d}_{k-1} - r_{k-1}\hat{d}_k, \tag{138}$$

where $\hat{d}_k$ is the receiver's estimate of the noiseless, perfectly-timed k^{th} sample d_k given by

$$d_k = \sum_l a_l h(kT - lT). \tag{139}$$

For the PR-IV channel that has the pulse shape

$$h(t) = \text{sinc}\left(\frac{t}{T}\right) - \text{sinc}\left(\frac{t-2T}{T}\right), \tag{140}$$

the noiseless, perfectly timed samples are given by

$$\begin{aligned} d_k &= \sum_l a_l h(kT - lT) \\ &= a_k - a_{k-2}. \end{aligned} \tag{141}$$

As discussed in Chapter 2, performance of the Mueller and Müller TED can be improved by using soft estimates $\tilde{d}_k$ in place of hard estimates $\hat{d}_k$. For the PR-IV channel, we use a memoryless soft slicer of the form

$$\tilde{d}_k = \frac{2\sinh(2r_k/\sigma^2)}{\cosh(2r_k/\sigma^2) + e^{2/\sigma^2}}. \tag{142}$$

4.2 PLL vs. CRB

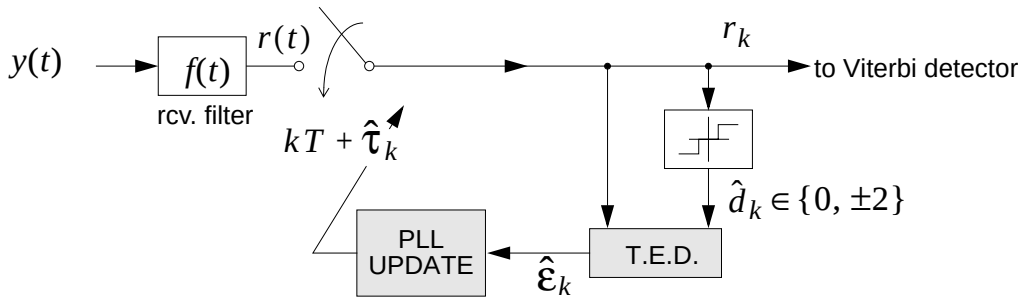

Figure 26: Conventional timing recovery.

The block diagram of the conventional timing recovery block is shown in Figure 26. The received waveform $y(t)$ is first filtered by a low-pass filter with impulse response $\mathrm{sinc}(t/T)$ and then sampled at instants $kT+\hat{\tau}_k$ determined by the PLL, which operates on the timing error estimates $\hat{\epsilon}_k$ produced by the TED. We use the MM TED with soft decisions of (142) as opposed to the conventionally used hard decisions. We next evaluate the performance of this scheme for the three timing models discussed earlier, namely constant offset, frequency offset and random walk.

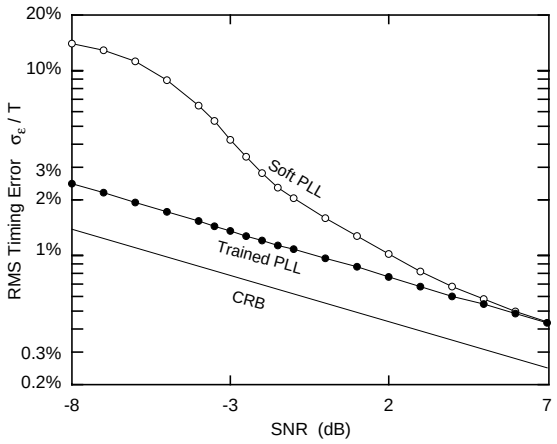

Figure 27: Constant Offset: Decision-directed case approaches the trained case as SNR increases.

4.2.1 Constant Offset

The (normalized) RMS timing error $\sigma_\epsilon = \sqrt{E[(\tau - \hat{\tau})^2]}$ is plotted in Figure 27 as a function of SNR for the PLL-based timing recovery system with channel impulse response $p(t) = \text{sinc}(t/T)$. The timing offset was chosen to be $\tau/T = \pi/20$, the block length $N = 5000$, $\alpha = 0.01$ and $\beta = 0$. To arrive at a single parameter $\hat{\tau}$ from the PLL output, we averaged the final 100 samples to leave out the initial transients, and also to have enough samples for the result to be statistically reliable. The gain parameters were chosen to minimize the timing error variance for the last 100 samples. As the SNR increases, the decisions become more reliable and the performance of the decision-directed system (denoted by "Soft PLL") approaches that of the trained PLL. The trained PLL is about 5 dB away from the CRB. This gap can be closed by using a variable-gain PLL. In the current set-up, the gain parameter α should be high enough to allow the PLL to rise to the actual value within the block length but low enough so that the estimate is not too noisy. But with the variable-gain PLL, we can choose a high value of α to start with to allow a small rise time and drop α to reduce the noise variance. In Chapters 5 and 7, we look at two other means of improving the performance:

- using better timing recovery architecture, and
- using better decisions in the TED through iterative timing recovery.

For the constant offset, we need to arrive at a single timing estimate parameter from the N parameters that form the output of the PLL. We restricted ourselves to timing estimates from the PLL towards the end of the packet. Another approach not explored here is to use a time-varying gain for the PLL, and combine the PLL outputs using some suitably chosen weights to arrive at a single timing estimate. This approach could only perform better than the one used for our simulation results.

4.2.2 Frequency Offset

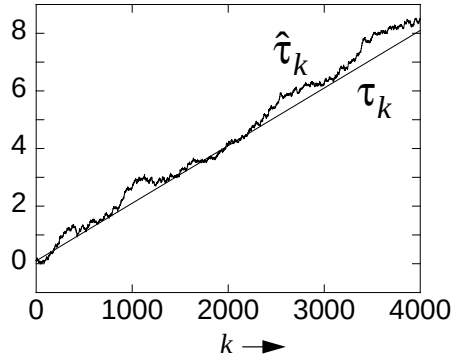

Figure 28: PLL timing estimate is a noisy version of the actual timing offset.

Conventional timing recovery uses a second-order PLL to track a frequency offset, and the PLL operates on timing error estimates generated by the Mueller and Müller TED as before. The output of the PLL, $\{\hat{\tau}_k\}$, is a noise-corrupted version of the actual timing offsets $\{\tau_k\}$. An example is shown in Figure 28. To extract the two parameters of interest $\hat{\tau}_0$ and $\hat{\Delta T}$ from the N outputs of the PLL $\{\hat{\tau}_k\}$, we use least squares estimation. Specifically,

$$\begin{aligned} \hat{\Delta T} &= \frac{\langle k\hat{\tau}_k\rangle - \langle k\rangle\langle\hat{\tau}_k\rangle}{\langle k^2\rangle - \langle k\rangle^2}, \\ \hat{\tau}_0 &= \langle\hat{\tau}_k - k\hat{\Delta T}\rangle, \end{aligned} \tag{143}$$

where $\langle x(k) \rangle = \frac{1}{N} \sum_{k=0}^{N-1} x(k)$.

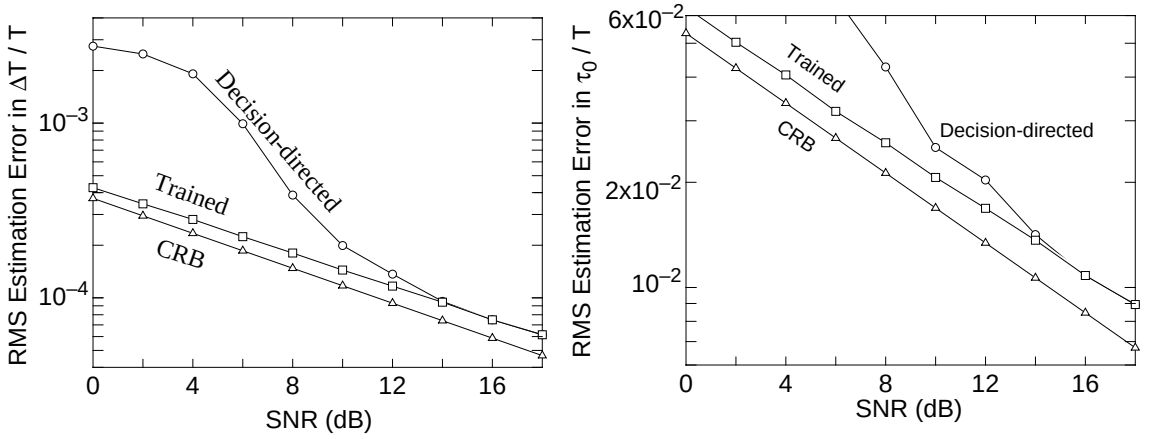

Figure 29: Frequency offset: Trained PLL-based system does not achieve the CRB.

Figure 29 shows the normalized RMS estimation error as a function of SNR for the PLL-based system with the PR-IV channel, averaged over 10000 blocks of length $N = 250$, with system parameters being $\Delta T/T \sim \text{unif}[0, 0.005]$, $\tau_0/T \sim \text{unif}[0, 0.1]$, α chosen to minimize the RMS estimation error, and $\beta = \alpha^2/4$. The trained system is about 2 dB away from the CRB. For low SNR, we have a significant performance penalty in the absence of training.

4.2.3 Random Walk

The performance of trained PLL gives a heuristic lower bound for the performance of realistic receiver structures that use the PLL for timing recovery. In Figure 30, we plot the steady state CRB and the RMS timing estimation error of the trained PLL with the PR-IV channel for the following system parameters: $\sigma_w/T = 0.5\%$, block length $N = 500$, and the PLL performance being averaged over 1000 trials. The performance of the PLL is a strong function of the gain parameter α, and therefore, α has to be optimized for each SNR. The PLL error variance is plotted for various values of α. Taking the minimum of the error variance over all α gives us the best performance we can expect using the trained PLL. We see that the average timing

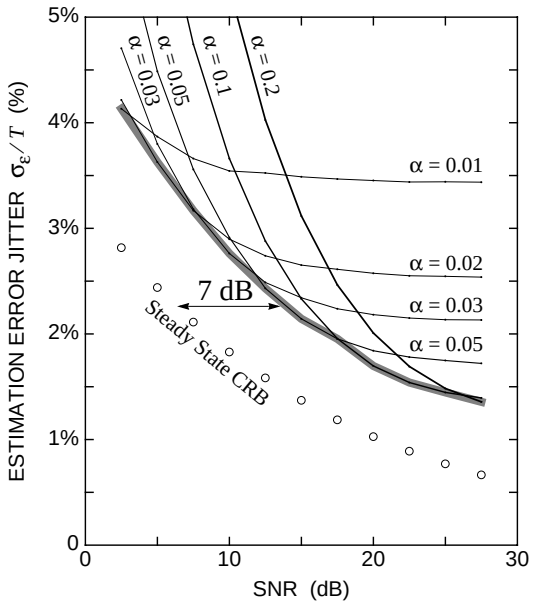

Figure 30: Random walk: Trained PLL does not achieve the steady-state CRB.

estimation error variance of the trained PLL is about 7 dB away from the steady-state CRB.

This gap of 7 dB has to be put in perspective by the fact that the CRB is not attainable in this case. For the CRB to be attainable, the *a posteriori* density $f_{\boldsymbol{\tau}|\boldsymbol{r}}(\boldsymbol{\tau}|\boldsymbol{r})$ needs to be Gaussian, where $\boldsymbol{r}$ is the vector of observations and $\boldsymbol{\tau}$ is the vector of timing parameters [63]. With the PR-IV channel and the random walk timing model, this is not the case.

Though the results above suggest that the PLL is not the optimal CRB-achieving strategy, a first-order PLL does minimize the timing estimation error variance with a random walk timing model and the causality constraint. Indeed, it turns out that a slightly modified first order PLL is the optimal (with respect to minimizing the timing estimation error variance) causal processing when tracking the general model involving an initial timing offset, frequency offset and a random walk. During acquisition, however, the optimal causal processing is a second-order PLL with time-varying gains. These results follow from applying adaptive Kalman filter theory to the problem of

tracking a random walk, which is discussed next.

4.3 PLL vs. Kalman Filter

Consider the general timing model with an initial timing offset τ_0 with *a priori* variance $\sigma^2_{\tau_0}$, a frequency offset parameter ΔT with *a priori* variance $\sigma^2_{\Delta T}$ and a random walk characterized by *i.i.d.* zero-mean normal random variables $\{w_k\}$ of variance σ^2_w. The evolution of the timing offsets can be modeled as [21] [12]

$$\begin{aligned} \boldsymbol{X}_{k+1} &= \begin{bmatrix} \tau_{k+1} \\ \dot{\tau}_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} \tau_k \\ \dot{\tau}_k \end{bmatrix} + \begin{bmatrix} w_k \\ \dot{w}_k \end{bmatrix} \\ &= \boldsymbol{F}\boldsymbol{X}_k + \boldsymbol{W}_k, \end{aligned} \tag{144}$$

For our timing model, $\dot{w}_k = 0$ and $\dot{\tau}_k = \Delta T$ for all k. This defines the state equation for the system. A simpler state equation would be

$$\tau_{k+1} = \tau_k + w_k, \tag{145}$$

where the random variables w_k are *i.i.d.* normal random variables with mean ΔT and variance σ^2_w. A difficulty with this model is that the mean of the random variables w_k is itself an unknown parameter, and is part of the estimation problem. While theoretically, there is little difference between the models of (144) and (145), the model of (144) allows standard Kalman filtering analysis. Therefore, we stick to the two-dimensional system characterization of (144).

We assume a timing error detector (TED) that generates timing error estimates $\hat{\epsilon}_k$ given by

$$\begin{aligned} \hat{\epsilon}_k &= \epsilon_k + \nu_k, \\ &= \tau_k - \hat{\tau}_k + \nu_k, \end{aligned} \tag{146}$$

where $\hat{\tau}_k$ is the receiver's estimate of τ_k, $\epsilon_k = \tau_k - \hat{\tau}_k$ is the timing error, and the observation noise terms ν_k are *i.i.d.* zero-mean normal random variables with variance

σ_v^2. This does not give an explicit measurement equation in terms of τ_k directly. We use a hypothetical measurement equation of the form

$$y_k = \tau_k + \nu_k. \tag{147}$$

Driessen [21] showed that the Kalman filter, which is the optimal causal processing to minimize the mean squared error for the linear Gaussian model, needs only $\hat{\epsilon}_k$ and generates timing estimates $\hat{\tau}_k$ according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha_k \hat{\epsilon}_k + \sum_{i=0}^{k-1} \beta_i \hat{\epsilon}_i, \tag{148}$$

which is exactly the structure of a second-order PLL, albeit with time-varying gains. Let $\hat{\boldsymbol{X}}_{k+1|k} = [\hat{\tau}_{k+1} \ \hat{\dot{\tau}}_{k+1}]^T$ be the Kalman filter estimate of $\boldsymbol{X}$ given all the observations from time 0 to time k. Then the Kalman update equation can be rewritten as

$$\hat{\boldsymbol{X}}_{k+1|k} = \boldsymbol{F}\hat{\boldsymbol{X}}_{k|k-1} + \boldsymbol{F}\boldsymbol{K}_k \hat{\epsilon}_k, \tag{149}$$

where $\boldsymbol{K}_k = [\alpha_k \ \beta_k]^T$ is the time-varying Kalman gain. Patapoutian [53] derived closed-form expressions for the gains in the so-called acquisition (*i.e.*, assuming $\sigma_w^2 = 0$) and the so-called tracking (*i.e.*, assuming $k \to \infty$) modes. The terminology of [53] regarding acquisition and tracking is based on the assumption of a slowly varying random walk that can be neglected during the short duration of the acquisition phase. In addition, only the asymptotic tracking performance was considered. In the so-called acquisition mode, the gain vector evaluates to [53]

$$\boldsymbol{K}_k = \frac{1}{\sigma_v^2 c_k} \begin{bmatrix} \frac{k^2\sigma_v^2}{\sigma_{\tau_0}^2} + \frac{\sigma_v^2}{\sigma_{\Delta T}^2} + \frac{k(K+1)(2K+1)}{6} \\ \frac{k\sigma_v^2}{\sigma_{\tau_0}^2} + \frac{k(k+1)}{2} \end{bmatrix}, \tag{150}$$

where

$$c_k = \frac{\sigma_v^2}{\sigma_{\tau_0}^2 \sigma_{\Delta T}^2} + \frac{k(k+1)(2k+1)}{6} + \frac{k+1}{\sigma_{\Delta T}^2} + \frac{k(k+1)^2(k+2)}{12\sigma_v^2}. \tag{151}$$

The estimation error covariance matrix $\boldsymbol{P}_{k|k-1} = E[(\boldsymbol{X}_k - \hat{\boldsymbol{X}}_{k|k-1})^2]$ is given by [53]

$$\boldsymbol{P}_{k|k-1} = \frac{1}{c_{k-1}} \begin{bmatrix} \frac{k^2\sigma_v^2}{\sigma_{\tau_0}^2} + \frac{\sigma_v^2}{\sigma_{\Delta T}^2} + \frac{k(k+1)(2k+1)}{6} & \frac{k\sigma_v^2}{\sigma_{\tau_0}^2} + \frac{k(k+1)}{2} \\ \frac{k\sigma_v^2}{\sigma_{\tau_0}^2} + \frac{k(k+1)}{2} & \frac{\sigma_v^2}{\sigma_{\tau_0}^2} + k \end{bmatrix}. \tag{152}$$

This is precisely the error variance corresponding to the CRB for estimating the timing parameters $\tau_0 + k\Delta T$ and ΔT assuming that we have only k data symbols. In Chapter 3, we derived the CRB for estimating τ_0 and ΔT. These can be modified for the parameters $\tau_0 + k\Delta T$ and ΔT, and the CRB in this case is the same as $\boldsymbol{P}_{k|k-1}$. Therefore, the Kalman filter is the optimal *causal* processing in that it achieves the CRB assuming that only the past observations are available. In Chapter 5, we develop a linear observation model for the timing offsets and analyze the optimal processing based on the linear model assuming that all the observations, and not just the ones from the past, are available.

In the so-called tracking mode of [53], the steady-state gain $\boldsymbol{K} = \lim_{k\to\infty} \boldsymbol{K}_k = [\alpha\ \beta]^T$ evaluates to [53]

$$\begin{aligned} \alpha &= 1 - z_0 z_1, \\ \beta &= (1 - z_0)(1 - z_1), \end{aligned} \tag{153}$$

where $z_i = (\gamma_i - \sqrt{\gamma_i^2 - 4})/2, i \in \{0, 1\}$ and

$$\gamma_i = 2 + \left(\frac{\sigma_w^2 - \sigma_{w\dot{w}}^2}{2\sigma_v^2}\right) + (-1)^i \sqrt{\left(\frac{\sigma_w^2 - \sigma_{w\dot{w}}^2}{2\sigma_v^2}\right)^2 - \frac{\sigma_{\dot{w}}^2}{\sigma_v^2}}. \tag{154}$$

In our case, $\sigma_{\dot{w}} = \sigma_{w\dot{w}} = 0$. This leads to

$$\begin{aligned} \alpha &= \frac{\sigma_w}{\sigma_v}\sqrt{1 + \frac{\sigma_w^2}{4\sigma_v^2}} - \frac{\sigma_w^2}{2\sigma_v^2}, \\ \beta &= 0. \end{aligned} \tag{155}$$

This means that the Kalman structure for tracking is simply a modified first-order PLL whose update equation is given by

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha\hat{\epsilon}_k + \hat{\Delta T}, \tag{156}$$

where $\hat{\Delta T}$ is the receiver's estimate of the frequency offset parameter ΔT. If the receiver has an independent means of estimating ΔT, then we simply need to run

a first-order PLL to track a random walk. This insight is used later in Chapter 6 where we choose the optimal preamble placement to minimize the CRB on timing estimation error variance.

4.4 Summary

In this chapter, we reviewed the conventional timing recovery method based on the PLL. The PLL updates the timing estimates based on the timing error estimates produced by the TED and the performance of the overall timing recovery block is a strong function of the performance of the TED. Next, we compared the performance of the PLL with the CRB for the PR-IV channel and observed that there is a gap between the performance of the PLL and the CRB. When tracking a random walk with a causality constraint at the receiver, we showed that the first-order PLL is the optimal timing recovery recovery method in that it achieves the CRB. In Chapter 5, we present block-processing, non-causal timing recovery architectures that perform better than the PLL. In Chapter 7, we present iterative timing recovery which is an alternative method of performance improvement by passing better decisions to the PLL when we have an outer error control code.

CHAPTER 5

OUTPERFORMING THE PLL

In Chapter 3, we derived lower bounds on the error variance of timing recovery algorithms. In Chapter 4, we compared the performance of the conventional PLL-based timing recovery scheme with these bounds and observed a performance gap. In this chapter, we consider alternatives to the PLL that perform better. As before, we treat four different cases: constant offset, frequency offset, random walk and the general timing model.

With a constant timing offset, trained maximum-likelihood (ML) estimation using gradient search achieves the CRB. With a frequency offset, simulations show that the trained Levenberg-Marquardt (LM) method [57], which is a combination of gradient descent and Newton's method, achieves the CRB. In both these cases, we use the PLL-based system as the initialization. The performance of both the gradient descent and the LM algorithm depends on the initialization provided, and without training especially at low SNR, the PLL performs poorly enough for these algorithms to not achieve the CRB.

With a random walk, ML estimation is prohibitively complex. Instead, we propose MAP timing recovery based on a linearization of the PLL output. MAP timing recovery takes the form of a matrix operation on the PLL outputs, and can be simplified to take the form of a time-invariant filtering followed by a time-varying scaling operation. With training, and for the system parameters considered here, this filtering-and-scaling approach performs 5 dB better than the PLL and is within 2 dB from the steady-state CRB.

The rest of the chapter is organized as follows. The system model under consideration is summarized in Section 5.1. Block processing algorithms for the constant offset, frequency offset and the random walk cases are presented in Sections 5.2, 5.3 and 5.4 respectively. Finally, we summarize results and observations in Section 5.5.

5.1 System Model

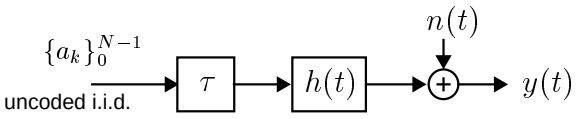

Figure 31: System block diagram with timing offsets, channel distortion and additive noise.

We again consider the system model described in Figure 31. The channel output waveform $y(t)$ is given by

$$y(t) = \sum_{l=0}^{N-1} a_l h(t - lT - \tau_l) + n(t), \tag{157}$$

where T is the bit period, $a_l \in \{\pm 1\}$ are the N *i.i.d.* data symbols, $h(t)$ is the channel impulse response, $n(t)$ is additive white Gaussian noise, and τ_l is the unknown timing offset for the l^{th} symbol.

After low-pass filtering $y(t)$ at the receiver to remove the out-of-band noise, the resulting continuous-time waveform $r(t)$ can be modeled as

$$r(t) = \sum_{l} a_l h(t - lT - \tau_l) + n_1(t), \tag{158}$$

where $\{a_k\}$ are the data symbols, $h(t)$ is the channel impulse response, $n_1(t)$ is band-limited to $[-\frac{1}{2T}, \frac{1}{2T})$, and $\{\tau_k\}$ are the timing offsets introduced by the channel.

The continuous-time waveform $r(t)$ is then sampled at time instants $\{kT + \hat{\tau}_k\}$ to arrive at samples $\{r_k\}$. If we choose $\hat{\tau}_k = 0$, we get uniform samples. For simplicity in

the sequel, we introduce the following notation. Let $\boldsymbol{r}$ be the vector of observations, $\boldsymbol{\tau}$ be the vector of timing parameters to be estimated and $\boldsymbol{a}$ be the vector of transmitted symbols.

5.2 Constant Offset: Maximum-likelihood Estimation

An estimator that achieves the CRB is called an *efficient* estimator. If an efficient estimator exists, then it is the maximum-likelihood (ML) estimator which maximizes the likelihood $f_{\boldsymbol{r}|\boldsymbol{\tau}}(\boldsymbol{r}|\boldsymbol{\tau})$, or equivalently, the *log likelihood function* $\ln f_{\boldsymbol{r}|\boldsymbol{\tau}}(\boldsymbol{r}|\boldsymbol{\tau})$ [63].

For the constant offset case, the waveform $r(t)$ is given by

$$r(t) = \sum_l a_l h(t - lT - \tau) + n_1(t), \tag{159}$$

and the ML estimate $\hat{\tau}$ is the one that minimizes the cost function $J(\hat{\tau}; \boldsymbol{a})$ given by

$$J(\hat{\tau}; \boldsymbol{a}) = \int_{-\infty}^{\infty} (r(t) - \sum_l a_l h(t - lT - \hat{\tau}))^2 \, \mathrm{d}t. \tag{160}$$

Assuming that $h(t)$ is band-limited, so that sampling at a rate of $1/T$ provides sufficient statistics, we can rewrite the cost function in terms of uniform samples $r_k = r(kT)$ as

$$J(\hat{\tau}; \boldsymbol{a}) = \sum_{k=-\infty}^{\infty} (r_k - \sum_l a_l h(kT - lT - \hat{\tau}))^2. \tag{161}$$

5.2.1 ML: Gradient Descent

For the trained case where $\boldsymbol{a}$ is known, the minimization of the cost function in (161) can be performed by gradient descent. The gradient search is initialized using an estimate $\hat{\tau}_0$ obtained by using a low-complexity method like a PLL. Further estimates are arrived at by using

$$\hat{\tau}_{i+1} = \hat{\tau}_i - \mu J'(\hat{\tau}_i; \boldsymbol{a}), \tag{162}$$

where μ is the learning constant and $J'(\hat{\tau}_i; \boldsymbol{a})$ is the gradient of $J(\hat{\tau}; \boldsymbol{a})$ (with respect to $\hat{\tau}$) evaluated at the current estimate $\hat{\tau}_i$.

Without training, the PLL in the initial step is operated in the decision-directed mode. For the gradient search, we use the cost function $J(\hat{\tau}; \hat{\boldsymbol{a}})$ where we have replaced the data symbols $\boldsymbol{a}$ with the receiver's estimates $\hat{\boldsymbol{a}}$.

At the i^{th} iteration ($i > 1$), new uniform samples r_k^{new} are generated by resampling $r(t)$ at time instants $kT + \hat{\tau}_i$. These are then used to generate memoryless soft estimates $\tilde{a}_k = E[a_k|r_k]$ which are used in place of a_k in (162). Resampling $r(t)$ might not be practical in many cases. We can instead use the fact that $\{r_k\}$ provides sufficient statistics for $r(t)$ and interpolate $\{r_k\}$ to get $\{r_k^{\text{new}}\}$ as follows:

$$r_k^{\text{new}} = \sum_l r_l \, \text{sinc}(kT - lT + \hat{\tau}_i). \tag{163}$$

5.2.2 Simulation

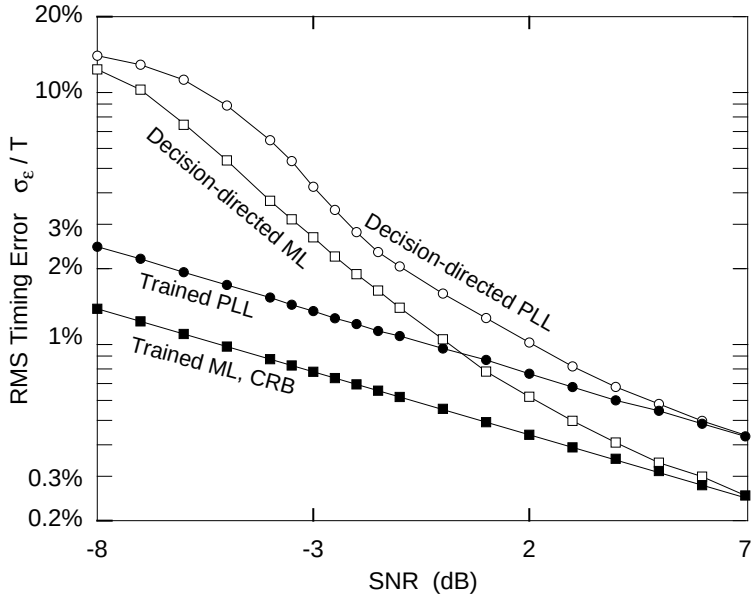

Figure 32: Constant Offset: ML estimator achieves the CRB.

The performance of the iterative ML estimator for the system with impulse response $h(t) = \text{sinc}(t/T)$ is shown in Figure 32 as a function of SNR. The algorithm is initialized with the PLL. The timing offset was chosen to be $\tau/T = \pi/20$, the block length $N = 5000$, and the PLL gains were $\alpha = 0.01$ and $\beta = 0$. These are the same

parameters chosen when we compared the performance of the PLL to the CRB for the constant offset case in Figure 27, and those curves are reproduced here. In addition, we have shown the trained and decision-directed ML cases in Figure 32. As opposed to the PLL case where we arrived at the timing estimate based on the final 100 PLL outputs in Section 4.2.1, the ML estimator directly outputs a single timing estimate, which is used in the RMS error computation. The trained ML appears to be efficient and the decision-directed case approaches the trained case as the decisions become more reliable.

5.3 Frequency Offset: ML Estimation

For the frequency offset case, the waveform $r(t)$ is given by

$$r(t) = \sum_l a_l h(t - lT - \tau_0 - l\Delta T) + n_1(t). \tag{164}$$

The ML estimator that produces estimates $\hat{\Delta T}$ and $\hat{\tau}_0$ minimizes the following cost function:

$$J(\hat{\tau}_0, \hat{\Delta T}; \boldsymbol{a}) = \int_{-\infty}^{\infty} (r(t) - \sum_l a_l h(t - lT - l\hat{\Delta T} - \hat{\tau}_0))^2 \, \mathrm{d}t. \tag{165}$$

Again, assuming that uniform samples $\{r_k\}$ taken at a sampling rate of $1/T$ provide sufficient statistics for $r(t)$, we can rewrite the cost function in terms of these samples as

$$J(\hat{\tau}_0, \hat{\Delta T}; \boldsymbol{a}) = \sum_{k=-\infty}^{\infty} (r_k - \sum_l a_l h(kT - lT - l\hat{\Delta T} - \hat{\tau}_0))^2. \tag{166}$$

As in the previous section, this minimization can be done using gradient descent. However, the cost function of (166) is not suited for gradient descent. To show this, in Figure 33, we plot the cost function $J(\hat{\tau}_0, \hat{\Delta T}; \boldsymbol{a})$ for a particular $\boldsymbol{a}$, ΔT and τ_0 as a function of $\hat{\Delta T}$ from $-0.1T$ to $0.1T$. In the graph, we have overlaid the cost function evaluated for values of $\hat{\tau}_0$ going from $-0.5T$ to $0.5T$. As a function of $\hat{\tau}_0$, the cost function is parabolic and therefore, gradient descent along $\hat{\tau}_0$ is feasible. But

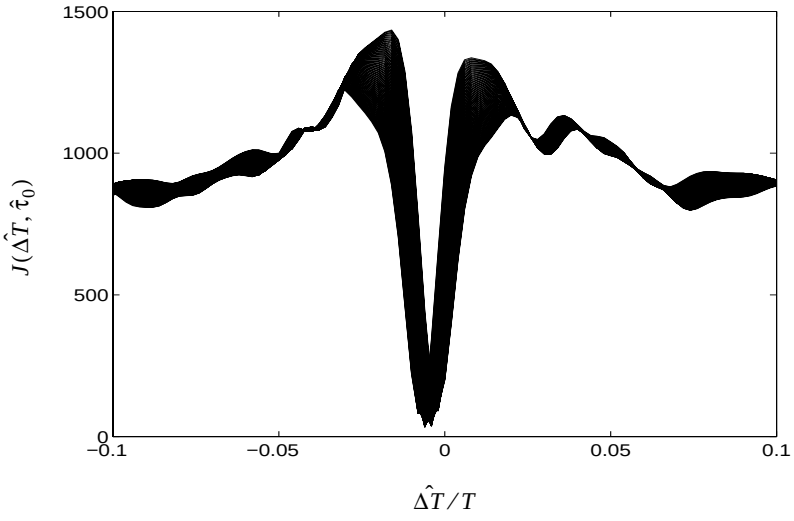

Figure 33: The cost function surface makes gradient descent unsuitable.

along $\hat{\Delta T}$, the global minimum of $J(\hat{\tau}_0, \hat{\Delta T}; \boldsymbol{a})$ occurs in a narrow valley. Hence, the efficacy of gradient search is very sensitive to initialization.

In the trained case, these issues can be addressed by implementing the Levenberg-Marquardt (LM) method, which is a combination of gradient descent and Newton's method [57]. The initial estimate for the LM iterations is obtained by performing a least-squares estimation using the output of the second-order PLL as described in Section 4.2.2. The LM method is then implemented as described below.

5.3.1 ML: Levenberg-Marquardt Method

Let $\boldsymbol{a} = [a_0\ a_1\ \ldots\ a_{N-1}]^T$ be the data vector, $\boldsymbol{\theta} = [\Delta T\ \tau_0]^T$ be the parameter vector, $\boldsymbol{r} = [r_0\ r_1\ \ldots\ r_{N-1}]^T$ where $r_k = \sum_l a_l h(kT - lT - l\Delta T - \tau_0)$ is the k^{th} noiseless uniform sample. Our model then becomes

$$\boldsymbol{r} = \boldsymbol{f}(\boldsymbol{a}; \boldsymbol{\theta}), \tag{167}$$

where $\boldsymbol{f}(\boldsymbol{a};\boldsymbol{\theta})$ is a suitably defined vector function. Denote the k^{th} element of $\boldsymbol{f}$ by f_k. Given $\boldsymbol{a}$ and noisy measurements $\hat{\boldsymbol{r}}$, we need to pick $\hat{\boldsymbol{\theta}}$ to minimize the error

$$E(\hat{\boldsymbol{\theta}}) = \sum_k (f_k(\boldsymbol{a};\hat{\boldsymbol{\theta}}) - \hat{r}_k)^2. \tag{168}$$

At the i^{th} iteration, let $\boldsymbol{\theta}_i$ be the current estimate. Implementing gradient descent, we would first compute the gradient $\boldsymbol{d}$ and then update the estimate to get $\boldsymbol{\theta}_{i+1}$ as follows.

$$\begin{aligned} \boldsymbol{d} &= \sum_k (f_k(\boldsymbol{a};\boldsymbol{\theta}_i) - \hat{r}_k)[\nabla \boldsymbol{f}(\boldsymbol{a};\boldsymbol{\theta}_i)]_k \\ \boldsymbol{\theta}_{i+1} &= \boldsymbol{\theta}_i - \mu \boldsymbol{d}. \end{aligned} \tag{169}$$

Here, $[\nabla \boldsymbol{x}]_k$ denotes the gradient of the k^{th} element of $\boldsymbol{x}$. Implementing Newton's method, we would compute the gradient $\boldsymbol{d}$ as above and the approximate Hessian $\boldsymbol{H}$, and then update as follows.

$$\begin{aligned} \boldsymbol{H} &= \sum_k [\nabla \boldsymbol{f}(\boldsymbol{a};\boldsymbol{\theta}_i)]_k [\nabla \boldsymbol{f}(\boldsymbol{a};\boldsymbol{\theta}_i)]_k^T \\ \boldsymbol{\theta}_{i+1} &= \boldsymbol{\theta}_i - \boldsymbol{H}^{-1}\boldsymbol{d}. \end{aligned} \tag{170}$$

Gradient descent works by moving along the direction of the greatest negative gradient, whereas Newton's method fits an approximate parabolic bowl to the surface and tries to locate the minimum of this bowl. These two methods can be combined by a weight factor λ which will determine the relative weight given to these two methods in the update equation:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - (\boldsymbol{H} + \lambda \boldsymbol{I})^{-1}\boldsymbol{d}. \tag{171}$$

The factor λ is adjusted according to whether the error E is increasing or decreasing at each iteration. If E has increased after the update, we retract the step, increase λ by a significant factor and try the update again. If the error has decreased, we accept the step, decrease λ by the same factor and proceed. Essentially, we put more weight in the Newton's method when we are in the right direction to converge faster, and revert to gradient descent if we are headed in the wrong direction.

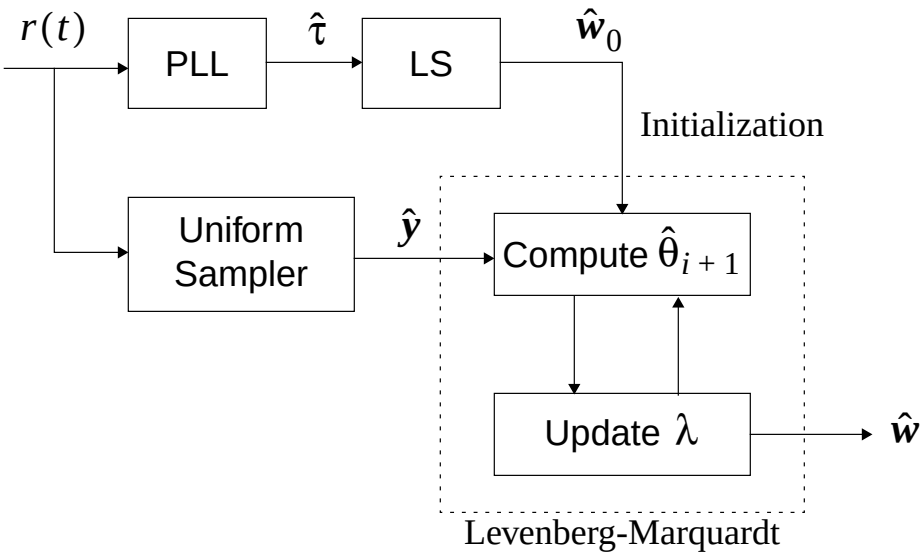

Figure 34: The Levenberg-Marquardt method.

To get around the problem of slow convergence in a long, narrow valley, we need to move in the direction in which the gradient is smaller, and this is achieved by replacing the identity in (171) with the diagonal of $\boldsymbol{H}$:

$$\boldsymbol{\theta}_{i+1} = \boldsymbol{\theta}_i - (\boldsymbol{H} + \lambda \text{diag}[\boldsymbol{H}])^{-1}\boldsymbol{d}. \tag{172}$$

This is the update equation for the Levenberg-Marquardt method. The LM algorithm is summarized in the block diagram shown in Figure 34.

5.3.2 Simulation

The LM algorithm was implemented on the PR-IV system with parameters $N = 250$, $\Delta T/T \sim \text{unif}[0, 0.005]$, and $\tau_0/T \sim \text{unif}[0, 0.1]$. For the PLL, α was chosen to minimize the RMS estimation error and $\beta = \alpha^2/4$. The RMS timing error for the PLL averaged over 10000 blocks was shown in Figure 29 and the PLL performed about 2 dB away from the CRB. In Figure 35, we plot the performance of the trained LM system which evidently achieves the CRB.

The LM algorithm speeds up convergence in the steep, narrow valley, but doesn't mitigate the sensitivity of the algorithm to the initialization. In the decision-directed

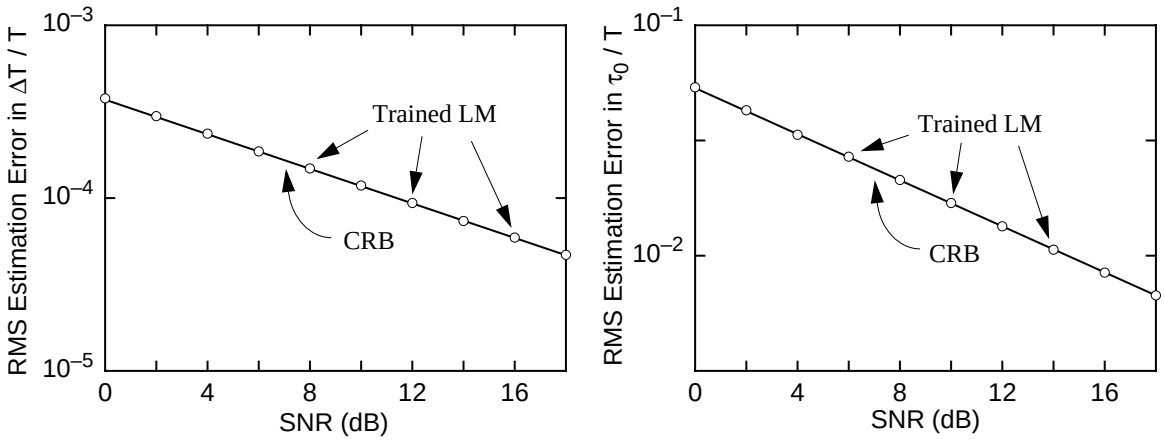

Figure 35: Frequency offset: Trained LM achieves the CRB.

case, the PLL performance is much worse without training, especially at low SNR. (Figure 29.) As we use the PLL for initialization, it is much more likely that the LM algorithm would converge to a local minimum that is not the global minimum. Without training, cycle slips occur much more often and therefore, the parameters $\hat{\Delta T}$ and $\hat{\tau}_0$ estimated using the least squares estimation from the PLL output become meaningless, further worsening the initialization. Therefore, we restrict the performance evaluation of the LM algorithm to the trained case.

5.4 Random Walk: Maximum A Posteriori (MAP) Estimation

In both the cases discussed so far, *i.e.*, the constant offset and the frequency offset cases, we assumed no *a priori* information about the parameters to be estimated. In this section, we present the random walk case where we have *a priori* information about the timing parameters. Implementing the ML estimator in this case is prohibitively complex. Instead, we develop a linear observation model based on the PLL output and then, for this linear system, we implement the MAP estimator.

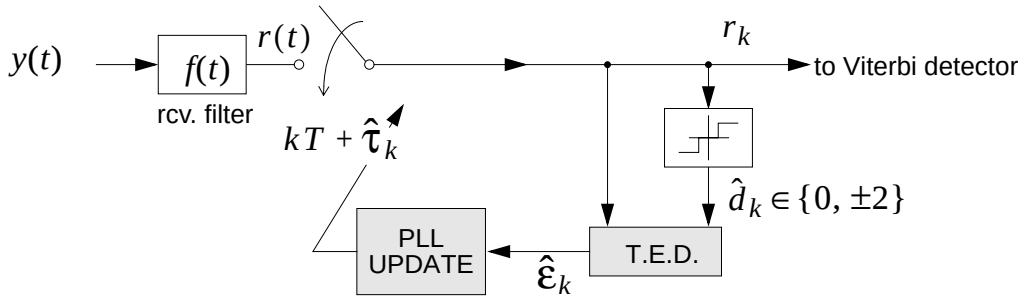

Figure 36: Conventional timing recovery.

5.4.1 Linear Model from TED and PLL

The PLL-based timing recovery architecture was discussed in Chapter 4. The block diagram is shown in Figure 36. The continuous-time waveform $r(t)$ is sampled at timing instants $kT + \hat{\tau}_k$ based on the estimates $\{\hat{\tau}_k\}$ of $\{\tau_k\}$ produced by the first-order PLL that updates its estimate of τ_k according to

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha\hat{\epsilon}_k, \tag{173}$$

where α is the PLL gain, and where $\hat{\epsilon}_k$ is the timing error detector's (TED) estimate of the estimation error $\epsilon_k = \tau_k - \hat{\tau}_k$. The Müller and Mueller (MM) TED generates $\hat{\epsilon}_k$ according to (138).

It was shown (Section 4.3, [53], [12]) that the PLL has the same structure as a Kalman filter that tracks the timing variations assuming a hidden linear measurement model

$$y_k = \tau_k + \nu_k, \tag{174}$$

where y_k is the implicit measurement and the noise ν_k is independent of $\{\tau_k\}$. Using the linearized model for the TED given by

$$\begin{aligned} \hat{\epsilon}_k &= \epsilon_k + \nu_k, \\ &= \tau_k - \hat{\tau}_k + \nu_k, \end{aligned} \tag{175}$$

we get the following simple method of accessing the hidden measurements y_k:

$$y_k = \hat{\tau}_k + \hat{\epsilon}_k. \tag{176}$$

The measurements y_k are, therefore, arrived at by adding corresponding outputs of the PLL and the TED. The PLL outputs an estimate of τ_k assuming that only observations $\{r_l\}_0^{k-1}$ are available. This is, therefore, an *a priori* estimate of τ_k. To this, we add $\hat{\epsilon}_k$, which is based on the present observation r_k as well. Therefore, y_k is an *a posteriori* estimate of τ_k and the quality of this estimate depends on the quality of the TED output. In the next section, we use these measurements $\{y_k\}$ to estimate $\{\tau_k\}$.

5.4.2 MAP Estimator for the Linear Model

In vector notation, the linear measurement model can be written as follows.

$$\boldsymbol{y} = \boldsymbol{\tau} + \boldsymbol{\nu}, \tag{177}$$

where

$$\begin{aligned} \boldsymbol{y} &= [y_0\ y_1\ \ldots\ y_{N-1}]^T, \\ \boldsymbol{\tau} &= [\tau_0\ \tau_1\ \ldots\ \tau_{N-1}]^T, \\ \boldsymbol{\nu} &= [\nu_0\ \nu_1\ \ldots\ \nu_{N-1}]^T. \end{aligned} \tag{178}$$

We assume that $\boldsymbol{\nu}$ is independent of $\boldsymbol{\tau}$, and also that $\boldsymbol{\nu}$ is $\mathcal{N}(\boldsymbol{0}, \sigma_n^2 \boldsymbol{I})$, where σ_n^2 is the error variance of the TED. To get the statistics of $\boldsymbol{\tau}$, consider the general case where the timing offsets consist of an initial phase offset ϕ_0, a frequency offset characterized by ΔT, and a random walk characterized by $\{w_k\}$, $k = 1, \ldots, N-1$.

Let $\boldsymbol{\phi} = [\Delta T \; \phi_0 \; w_1 \; \ldots \; w_{N-1}]^T$. Then

$$\boldsymbol{\tau} = \begin{bmatrix} 0 & 1 & & & & \\ 1 & 1 & 1 & & & \\ 2 & 1 & 1 & 1 & & \\ \vdots & \vdots & \vdots & \vdots & \ddots & \\ N-1 & 1 & 1 & 1 & \cdots & 1 \end{bmatrix} \boldsymbol{\phi} = \boldsymbol{M}\boldsymbol{\phi}. \tag{179}$$

The vector $\boldsymbol{\phi}$ is assumed to be $\mathcal{N}(\mathbf{0}, \boldsymbol{K}_{\boldsymbol{\phi}})$ where $\boldsymbol{K}_{\boldsymbol{\phi}} = \text{diag}(\sigma^2_{\Delta T}, \sigma^2_{\phi_0}, \sigma^2_w, \ldots, \sigma^2_w)$. Therefore, $\boldsymbol{\tau}$ is Gaussian as well, with a covariance matrix given by [63]

$$\boldsymbol{K}_{\boldsymbol{\tau}} = \boldsymbol{M}\boldsymbol{K}_{\boldsymbol{\phi}}\boldsymbol{M}^T. \tag{180}$$

Evaluating this, we get

$$[\boldsymbol{K}_{\boldsymbol{\tau}}]_{ij} = ij\sigma^2_{\Delta T} + \sigma^2_{\phi_0} + \min(i,j)\sigma^2_w, \tag{181}$$

for $i, j = 0, 1, \ldots, N-1$.

The MAP estimator for this linear Gaussian model is [63]

$$\hat{\boldsymbol{\tau}}_{\text{map}}(\boldsymbol{y}) = (\boldsymbol{K}_{\boldsymbol{\tau}} + \sigma^2_n \boldsymbol{I})^{-1}\boldsymbol{K}_{\boldsymbol{\tau}}\boldsymbol{y}. \tag{182}$$

An alternative representation of this estimator is using the eigendecomposition of $\boldsymbol{K}_{\boldsymbol{\tau}}$. Let $\{\boldsymbol{W}, \boldsymbol{\Lambda}\}$ represent this decomposition, such that $\boldsymbol{K}_{\boldsymbol{\tau}} = \boldsymbol{W}^T\boldsymbol{\Lambda}\boldsymbol{W}$, where $\boldsymbol{\Lambda}$ is a diagonal matrix with the main diagonal entries being $\{\lambda_i\}_{i=1}^N$, the N eigenvalues of $\boldsymbol{K}_{\boldsymbol{\tau}}$, and $\boldsymbol{W}$ is a unitary matrix with the i^{th} row being the eigenvector of $\boldsymbol{K}_{\boldsymbol{\tau}}$ corresponding to the eigenvalue λ_i. The MAP estimator can then be rewritten as

$$\hat{\boldsymbol{\tau}}_{\text{map}}(\boldsymbol{y}) = \boldsymbol{W}^T(\boldsymbol{\Lambda} + \sigma^2_n \boldsymbol{I})^{-1}\boldsymbol{\Lambda}\boldsymbol{W}\boldsymbol{y}. \tag{183}$$

The MAP estimator for this system is efficient, *i.e.*, it achieves the CRB for the linear model of (177). Let $\boldsymbol{\epsilon} = \boldsymbol{\tau} - \hat{\boldsymbol{\tau}}_{\text{map}}(\boldsymbol{y})$ be the estimation error. The error variance for the MAP estimator is given by

$$E[\boldsymbol{\epsilon}^T\boldsymbol{\epsilon}] = \sigma^2_n \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_i + \sigma^2_n}. \tag{184}$$

The error variance of the MAP estimator is a function of the TED noise variance σ_n^2. In Figure 7 of Chapter 2, we plotted the second moment of the MM TED timing error as a function of the actual timing error ϵ for SNR = 10 dB for the PR-IV system. The second moment is not independent of ϵ. Also, the estimator is not unbiased over the entire range of ϵ. However, to simplify the analysis, we assume unbiasedness of the TED. Also, we pick the value of the timing error second moment at the origin as σ_n^2, *i.e.*, $\sigma_n^2 = E[\hat{\epsilon}^2]_{\epsilon=0}$. This is a good approximation to the actual performance in the tracking mode where ϵ/T is small, and also gives a lower bound on the actual performance since $E[\hat{\epsilon}^2]_{\epsilon=0} < E[\hat{\epsilon}^2]_{\epsilon\neq 0}$.

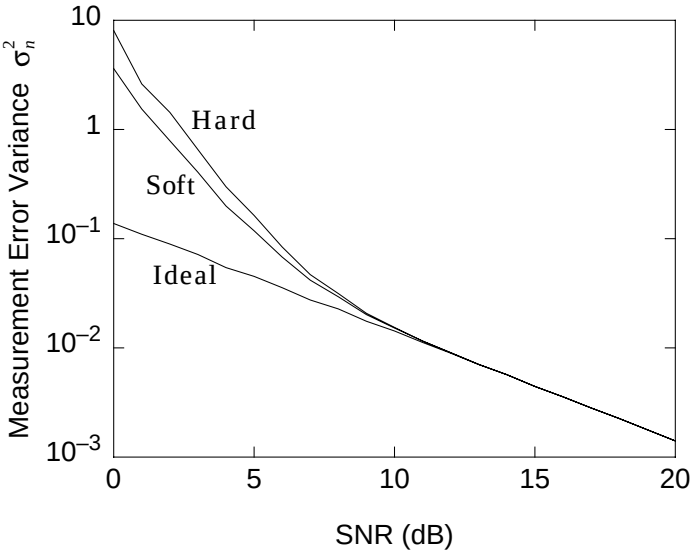

Figure 37: MM TED measurement error variance.

Figure 37 plots the measurement error variance σ_n^2 for the MM TED with ideal, hard and soft (both memoryless) decisions. As SNR increases, both hard and soft decisions become more accurate, and the error variance approaches that of the ideal decision case.

Figure 38 compares the performance of the MAP estimator with the trained PLL performance and the steady-state CRB for a random walk timing offset model with $\sigma_w/T = 0.33\%$, averaged over 1000 blocks of length $N = 500$. The MAP estimator

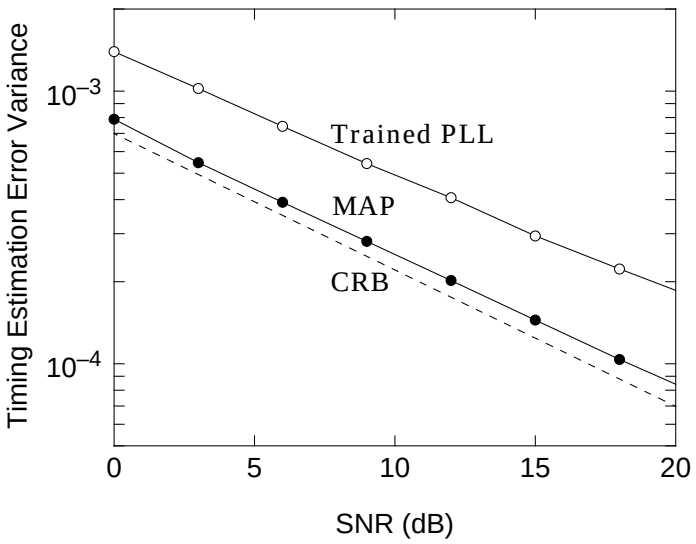

Figure 38: Actual MAP performance 1.5 dB away from the steady-state CRB.

operated on the output of a trained PLL. The MAP estimator performs to within 1.5 dB of the CRB and is around 5.5 dB better than the trained PLL. This 1.5 dB gap has to be put in perspective by the fact that the CRB is not attainable for the overall non-linear system with a PR-IV channel and a random walk timing model (Section 4.2.2). An additional contributor to the performance gap is the inaccuracy of the linear model.

The MAP estimator developed here can be viewed in a more general context. For any non-linear system characterized by a non-linear measurement equation, the traditional approach has been to linearize the measurement equation using Taylor series techniques and then use the Kalman filtering approach. The approach taken here is an alternative approach. With a suitable chosen *timing error detector*, the *phase-locked loop* gives us access to linear observations and we could then use the projection operator defined by the MAP equation to get the desired estimates.

5.4.3 Suboptimal Low-Complexity Implementations

The MAP estimator described above involves matrix operations and becomes infeasible or at least computationally burdensome for reasonable block lengths of around

5000 which are common in the magnetic recording industry. In this section, we propose suboptimal implementations of the MAP estimator that allow a trade-off between performance and complexity.

We first look at the MAP estimator error covariance matrix $\boldsymbol{K}_{\boldsymbol{\epsilon}} = E[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]$. In terms of the eigendecomposition of $\boldsymbol{K}_{\boldsymbol{\tau}}$, this evaluates to

$$\begin{aligned}\boldsymbol{K}_{\boldsymbol{\epsilon}} = \quad & \boldsymbol{W}^T \{(\boldsymbol{\Lambda} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{\Lambda} (\boldsymbol{\Lambda} + \sigma_n^2 \boldsymbol{I}) \boldsymbol{\Lambda} (\boldsymbol{\Lambda} + \sigma_n^2 \boldsymbol{I})^{-1} \cdots \\ & -(\boldsymbol{\Lambda} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{\Lambda}^2 - \boldsymbol{\Lambda}^2 (\boldsymbol{\Lambda} + \sigma_n^2 \boldsymbol{I})^{-1} + \boldsymbol{\Lambda} \Big\} \boldsymbol{W}. \end{aligned} \tag{185}$$

Recognizing that $\boldsymbol{\Lambda}$ is a diagonal matrix, this can be simplified to

$$\boldsymbol{K}_{\boldsymbol{\epsilon}} = \sigma_n^2 \boldsymbol{W}^T (\boldsymbol{\Lambda} + \sigma_n^2 \boldsymbol{I})^{-1} \boldsymbol{\Lambda} \boldsymbol{W}. \tag{186}$$

From (183) and (186), we can write

$$\hat{\boldsymbol{\tau}}_{\text{map}}(\boldsymbol{y}) = \frac{1}{\sigma_n^2} \boldsymbol{K}_{\boldsymbol{\epsilon}} \boldsymbol{y}. \tag{187}$$

Let $\boldsymbol{f_1}$ be the $N \times 1$ vector containing the elements of the main diagonal of $\boldsymbol{K}_{\boldsymbol{\epsilon}}$ and let $\boldsymbol{f} = \boldsymbol{f_1}/(\boldsymbol{f_1}(N/2))$. (We have assumed that N is even. For odd N, pick $\lfloor N/2 \rfloor$.) We call $\boldsymbol{f}$ the shaping function. Figure 39 shows the shaping function for $\sigma_w/T = 0.33\%$, SNR = 10 dB, and $N = 500$.

Figure 40 plots row 250 of $\boldsymbol{K}_{\boldsymbol{\epsilon}}$ with same parameters as above. In general, let $\boldsymbol{g} = [g_1 \ g_2 \ \cdots \ g_N]$ represent the $(N/2)^{th}$ row of $\boldsymbol{K}_{\boldsymbol{\epsilon}}$. (We have assumed that N is even. For odd N, pick $\lfloor N/2 \rfloor$.) The other rows of $\boldsymbol{K}_{\boldsymbol{\epsilon}}$ are very well approximated by suitably zero-padding and shifting $\boldsymbol{g}$, and scaling it by the corresponding element of the shaping function. For instance, denote the i^{th} row of $\boldsymbol{K}_{\boldsymbol{\epsilon}}$ by row_i, and let $i < N/2$. Let $\text{shift}(\boldsymbol{x}, k, \text{dir})$ denote the operator whose output is the vector formed by suitably zero-padding the vector $\boldsymbol{x}$ and shifting it by k units to the left or to the right depending on whether dir = left or dir = right respectively. Then

$$\text{row}_i \approx f(i) \times \text{shift}(\boldsymbol{g}, \frac{N}{2} - i, \text{left}). \tag{188}$$

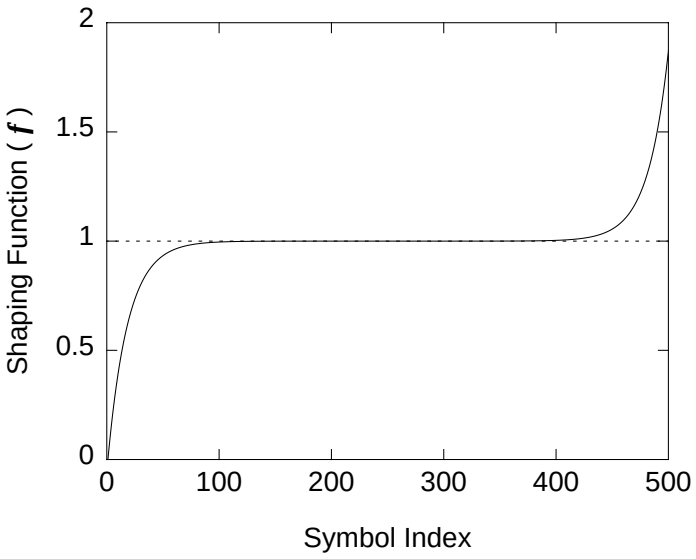

Figure 39: The shaping function has steady-state value of unity.

The case for $i > N/2$ is similar.

Essentially, we have approximated the MAP operation by the following structure.

$$\hat{\boldsymbol{\tau}}_{\text{map}}(\boldsymbol{y}) \approx \boldsymbol{A}_1 \boldsymbol{A}_2 \boldsymbol{y}, \tag{189}$$

where $\boldsymbol{A}_1$ is a diagonal matrix with the i^{th} main diagonal entry being $f(i)$ and $\boldsymbol{A}_2$ is the matrix whose rows are the shifted rows defined by (188) neglecting the multiplicative factor $f(i)$. This simplifies the implementation greatly since $\boldsymbol{A}_2$ represents a convolution matrix, and can be implemented as a time-invariant filter whose impulse response is $\boldsymbol{g}$. $\boldsymbol{A}_1$ can be implemented as time-varying scaling of the filter output. The approximate MAP estimator is shown in Figure 41.

To reduce complexity further, we can neglect the factor $\boldsymbol{A}_1$ altogether, and also truncate the filter to a manageable length. The losses associated with these are shown in Figure 42 for a random walk timing offset model with $\sigma_w/T = 0.33\%$, and $N = 500$. The approximation of (189) performs as well as the matrix operation of (183) leading to significant reduction in the memory reduction. As opposed to the N^2 memory elements needed for the matrix, we now need only $2N$ elements. Neglecting the shaping function and implementing only the filtering reduces the memory requirement to N

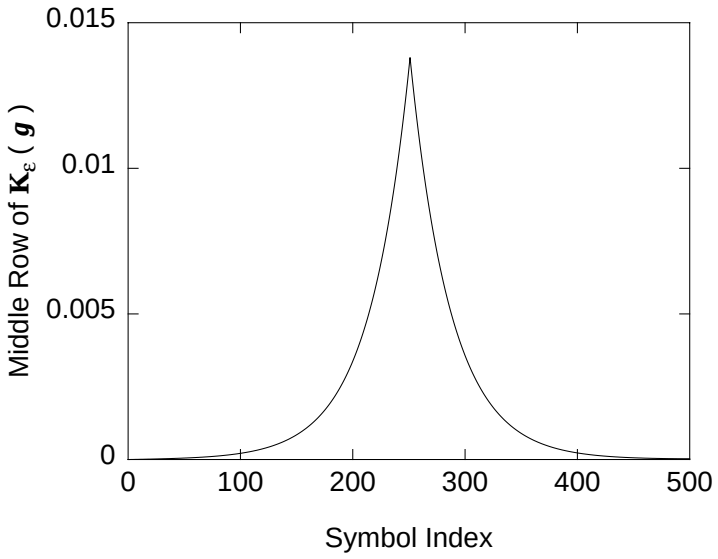

Figure 40: Except for shifting and scaling, this is a typical row of $\boldsymbol{K}_{\boldsymbol{\epsilon}}$.

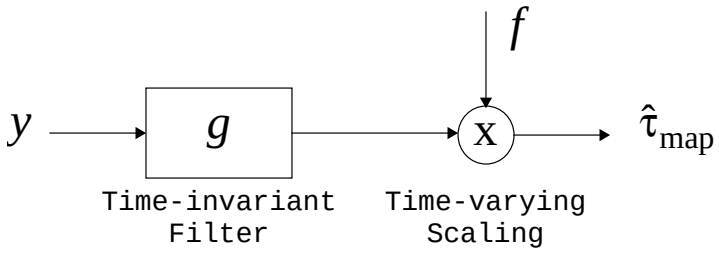

Figure 41: Approximate MAP estimator.

but more importantly allows a time-invariant filter-implementation. This approximation leads to a 1.5 dB loss. Further reduction in complexity by truncating the filter to a total of 100 terms as opposed to 500 leads to a further loss of less than 0.5 dB at $\sigma_\epsilon^2/T^2 = 2 \times 10^{-4}$, still 5 dB better than the PLL. Also, this approximation shows a cross-over with respect to the performance of the PLL for SNR around 5 dB, *i.e.*, for SNR $<$ 5dB, this performs worse than the PLL. This cross-over threshold varies with the number of filter coefficients used.

All the suboptimal implementations used above required the computation of the error covariance matrix $\boldsymbol{K}_{\boldsymbol{\epsilon}}$. Once we have $\boldsymbol{K}_{\boldsymbol{\epsilon}}$, instead of multiplying the vector $\boldsymbol{y}$ by $\boldsymbol{K}_{\boldsymbol{\epsilon}}$, we performed filtering and scaling to reduce complexity. The next simplifying

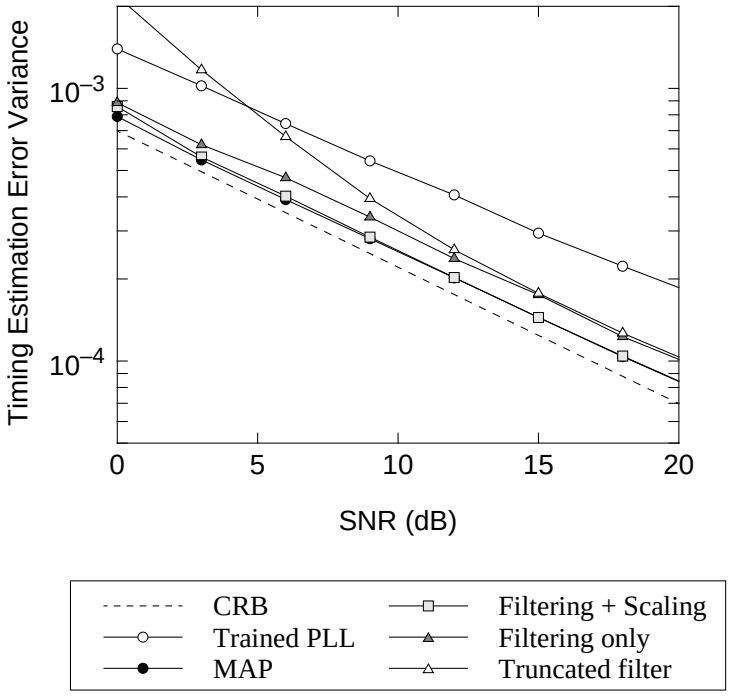

Figure 42: Various suboptimal strategies.

approximation allows us to analytically arrive at the filtering and the scaling values. We note that the MAP estimator performs close to the CRB and therefore, we replace $\boldsymbol{K_\epsilon}$ in the estimator equation by $\boldsymbol{K_{CRB}}$ as follows.

$$\hat{\boldsymbol{\tau}}\text{map}(\boldsymbol{y}) = \frac{1}{\sigma_n^2}\boldsymbol{K_{CRB}}\boldsymbol{y}. \tag{190}$$

With this approximation, we use the CRB expression of Chapter 3, (115), to get

$$\begin{aligned} f(i) &= \frac{1-\frac{\eta^{N-2i}-\eta^{-N+2i+3}}{\eta^{N+2}-\eta^{-N+1}}}{1-\frac{1-\eta^3}{\eta^{N+2}-\eta^{-N+1}}}, \\ g(i) &= \frac{\sigma_w^2}{\eta^2-1}\,\frac{\eta^{\frac{3N}{2}-i+2}-\eta^{\frac{N}{2}-i}+\eta^{-\frac{N}{2}+3+i}-\eta^{-\frac{3N}{2}+1+i}}{\eta^{N+1}-\eta^{-N}}, \end{aligned} \tag{191}$$

where

$$\eta = \frac{\lambda+\sqrt{\lambda^2-4}}{2} \quad \text{and} \quad \lambda = 2+E_{h'}\frac{\sigma_w^2}{\sigma^2}. \tag{192}$$

Here we have assumed that we have only a random walk timing offset. For the general model, we use the appropriate CRB derived in Chapter 3. This approximation does

not lead to any perceptible difference in performance in the suboptimal strategies discussed earlier. One more advantage with this analytical approach is that we need not experimentally measure the second moment of the timing error of the TED. The expressions are functions only of the channel SNR and the random walk parameter.

5.5 Summary

We proposed methods of achieving the CRB where possible, and where this was not possible, we proposed methods to improve on the performance of the conventional PLL-based approach. With a constant timing offset, gradient descent appears to achieve the CRB in the trained case. With a frequency offset, the cost function to be minimized exhibits a long, narrow valley and therefore, we use a combination of gradient descent and Newton's method to speed up convergence. However, this combination is rather sensitive to the initialization used. In the trained case, this method evidently achieves the CRB. With a random walk, we proposed MAP timing recovery, where we performed post-processing on the outputs of the PLL and the TED. This performs to within 1.5 dB of the CRB for the system parameters considered. The MAP algorithm takes the form of a matrix operation on the TED and the PLL outputs, which becomes impractical for reasonable block lengths. Therefore, we proposed suboptimal low-complexity implementations of the MAP algorithm, based on approximating it as a time-invariant filter followed by time-varying scaling.

The main drawback of all the schemes presented in this chapter is the fact that they are all based on the PLL-based system for initialization, and without training and at low SNR, are vulnerable to cycle slips. In Chapter 6, we present preamble placement strategies to reduce cycle slip occurrence, which in turn improves the performance of these algorithms. In contrast, iterative timing recovery, presented in Chapter 7, does not suffer from this problem even though it is based on the PLL since it is able to exploit the presence of the ECC and *correct* cycle slips on its own.

CHAPTER 6

ACQUISITION

In the previous chapters, we developed bounds on timing estimation and also developed timing recovery methods that outperform the conventional PLL-based ones. However, we have implicitly assumed knowledge of when the actual data transmission started. Usually, this knowledge is obtained during the acquisition phase. This is in contrast to the tracking phase, where the knowledge from the acquisition phase is fine-tuned and timing variations are tracked. Acquisition can be thought of as being part of the tracking problem with the initial offset now being a large offset and the frequency offset too being unknown. It is profitable, however, to separate the task of timing recovery into two phases. During the first phase, we use trained data to get a coarse estimate of the initial offset and the frequency offset and during the second phase, we fine-tune these estimates and track timing variations like the random walk. During this second phase, we do not have training. The advantage with this approach is reduced complexity of the estimator since it is easier and less complex to design tracking algorithms that estimate smaller timing offsets. The acquisition phase gets a coarse estimate fast, though with not too much accuracy, and the tracking algorithms can fine-tune these estimates.

In this chapter, we study a common method of acquisition which is based on the use of a preamble sequence. Conventionally, the preamble sequence is placed at the start of the packet. We add an additional degree of freedom to the acquisition problem by assuming that the position of the preamble need not be fixed and show that, to minimize the CRB on frequency offset estimation, we need to split the available known symbols into two halves and place them one each at the beginning and at the end of

the packet. For some specific cases, we show that this placement also minimizes the CRB when we perform joint acquisition and tracking. An added advantage of this placement strategy is that it greatly reduces the occurrence of cycle slips in the case when the frequency offset is the dominant timing offset.

The rest of the chapter is organized as follows. In Section 6.1, we present a brief description of previous work related to preamble placement optimization, which was with reference to channel estimation as opposed to timing recovery. In Section 6.2, we review the conventional timing acquisition method. Next, in Section 6.3, we present the linear observation model that is used to derive the CRB and also to arrive at the optimal preamble placement strategy. In Section 6.4, we evaluate the performance of the placement strategy derived in the previous section for the actual non-linear system with the PR-IV channel model, and observe that the CRB-achieving preamble placement strategy is the same. In Section 6.5, we simulate the PLL-based system with the CRB-achieving placement strategy and observe a reduction in the frequency of cycle slips. In Section 6.6, we consider joint acquisition and tracking. Finally, we summarize the results and observations in Section 6.7.

6.1 History: Preamble Placement for Channel Estimation

The problem of optimal preamble placement has been considered in literature with respect to channel estimation [1] [49] [20] [36] [10]. Different optimization criteria considered are the mean squared channel estimation error and the channel *i.i.d.* capacity.

The channels considered are finite impulse response fading channels, and the optimal training strategy consists of arranging the known symbols is small contiguous clusters placed periodically in the data stream [1] [49]. These preamble clusters are chosen to be as *equal* as possible, subject to the power constraint on training, and also

to number of taps of the channel impulse response. In [20], it is shown that superimposed training outperforms the time-division approach to training. In other words, it is shown that, for a given fraction of the transmit power allocated to training, it is beneficial to have a training signal component superimposed on the data signal, as opposed to inserting training symbols between data symbols. In [36], channels that are both time and frequency selective are considered. It is shown that to minimize the mean squared channel estimation error, the optimal training strategy is to use equi-spaced and equi-powered pilot symbols.

In [10], multiple access communication systems undergoing fading are considered. To minimize the influence of asynchronous interference on the packets of one user from those of other users, it is shown that the best placement strategy is to use two clusters of equal or quasi-equal size at the ends of the packet.

In this chapter, we consider the problem of optimal preamble placement to minimize the Cramér-Rao bound on the timing estimation error variance. This is similar to minimum mean-squared error approach considered in the channel estimation case. First, we begin by giving a description of the conventional acquisition method.

6.2 Conventional Acquisition

In a conventional setting, a certain number of known symbols are placed at the start of each packet to aid acquisition. At the receiver, the incoming waveform is correlated with the waveform generated based on the known data and the channel model, and peak detection is used to detect the start of transmission. The symbols chosen for this section of the preamble have to be such that we get very high correlation when the waveforms match and the correlation should die down rapidly as the waveforms get mismatched. The goal of this phase of acquisition is to get to within $[-0.5T, 0.5T)$ of the actual start of transmission, where T is the symbol duration. Depending on the channel and the noise conditions, we can do better than just get in the $[-0.5T, 0.5T)$

range and, often, the residual timing error can be effectively modeled as a zero-mean Gaussian random variable. A popular method for this phase of acquisition is the zero-phase start (ZPS) method [56].

The next portion of the preamble is often a periodic pattern that helps in acquiring the frequency offset. The periodic tone is spectrally compact and therefore, passes through the channel without distortion and can be detected easily at the receiver. A common method is to use a trained PLL at the receiver. The trained PLL will operate satisfactorily whenever the remaining offset it faces is small. The PLL gain is set reasonably high to allow for fast frequency acquisition. High gain increases the frequency range over which we can lock in and also reduces the overhead by reducing the required length of the preamble. At the end of this phase of acquisition, the residual frequency offset can also be modeled as a zero-mean Gaussian random variable.

6.3 Linear Model: Optimal Preamble Placement

As described in the earlier section, in the conventional acquisition set-up, all the known symbols are placed at the start of the packet. In this section, we relax this constraint and allow the placement of the known symbols to be an additional degree of freedom and study the improvements to be gained as a result of this relaxation. As before, during acquisition, we are interested in estimating only the initial timing offset and the frequency offset. We start with the linear measurement model developed in Chapter 5 where we added the outputs of the TED and the PLL to arrive at the observations y_k. We concentrate our attention on the second phase of acquisition where we operate the trained PLL to acquire the frequency. We analyze acquisition performance for the linearized system as a function of the preamble position and arrive at the optimal placement that minimizes the timing estimation error variance.

The linear model is given by

$$\boldsymbol{y} = \boldsymbol{\tau} + \boldsymbol{\nu}, \tag{193}$$

where

$$\begin{aligned} \boldsymbol{y} &= [y_0\ y_1\ \ldots\ y_{K-1}]^T, \\ \boldsymbol{\tau} &= [\tau_0\ \tau_1\ \ldots\ \tau_{K-1}]^T, \\ \boldsymbol{\nu} &= [\nu_0\ \nu_1\ \ldots\ \nu_{K-1}]^T. \end{aligned} \tag{194}$$

The noise variables $\{\nu_k\}$ are *i.i.d.* zero-mean normal random variables of variance σ_n^2. We assume that K out of the N data symbols are known. Therefore, for this model, we have K observations out of the total allowed N, and this is done by selecting only the observations corresponding to the acquisition slots.

The timing offsets τ_k themselves are from the following linear model.

$$\tau_k = \tau_0 + x_k \Delta T, \tag{195}$$

where x_k is the position of the k^{th} acquisition symbol, chosen from 0 to $N-1$. Therefore, the problem of choosing the preamble placement strategy is the same as choosing the set $\{x_k\}_0^{K-1}$.

6.3.1 Least Squares Estimation

The least squares estimator for the parameters ΔT and τ_0 is given by

$$\begin{aligned} \hat{\Delta T} &= \frac{K\sum x_m y_m - \sum x_m \sum y_m}{K\sum x_m^2 - (\sum x_m)^2}, \\ \hat{\tau_0} &= \frac{1}{K}\sum (y_m - x_m \hat{\Delta T}). \end{aligned} \tag{196}$$

For *i.i.d* observations $\{y_k\}$, this estimator is efficient, i.e., it achieves the Cramér-Rao bound and the estimation error variances are

$$\begin{aligned} V_{\Delta T} &= \frac{\sigma_n^2 K}{K\sum x_m^2 - (\sum x_m)^2}, \\ V_{\tau_0} &= \frac{\sigma_n^2 \sum x_m^2}{K\sum x_m^2 - (\sum x_m)^2}, \end{aligned} \tag{197}$$

where σ_n^2 is the variance of each noise variable ν_k. These expressions can be rewritten as

$$\begin{aligned} V_{\Delta T} &= \frac{\sigma_n^2}{K}\frac{1}{\frac{1}{K}\sum x_m^2 - (\frac{1}{K}\sum x_m)^2} &= \frac{\sigma_n^2}{K}\frac{1}{\mathrm{var}(\boldsymbol{I})}, \\ V_{\tau_0} &= \frac{\sigma_n^2}{K}\frac{\frac{1}{K}\sum x_m^2}{\frac{1}{K}\sum x_m^2 - (\frac{1}{K}\sum x_m)^2} &= \frac{\sigma_n^2}{K}\left(1 + \frac{\mathrm{mean}(\boldsymbol{I})^2}{\mathrm{var}(\boldsymbol{I})}\right), \end{aligned} \tag{198}$$

where $\boldsymbol{I}$ is the vector of acquisition indices $\{x_k\}$. In addition, we have defined the mean and variance of any given vector $\boldsymbol{S} = [s_0, s_1, \cdots, s_{K-1}]$ to be $\mathrm{mean}(\boldsymbol{S}) = \frac{1}{K}\sum s_k$ and $\mathrm{var}(\boldsymbol{S}) = \frac{1}{K}\sum s_k^2 - \left(\frac{1}{K}\sum s_k\right)^2$. The problem of choosing the best preamble locations, therefore, is the same as choosing the set $\boldsymbol{I} = \{x_k\}_0^{K-1}$.

First, we arrive at the optimal indices to minimize $V_{\Delta T}$ for fixed K and σ_n^2, which is the same as choosing the vector $\boldsymbol{I}$, the elements of which have the maximum variance.

6.3.2 Choosing the optimal $\boldsymbol{I}$

Theorem: Let K be an even natural number. Let $\boldsymbol{I} = [i_0, i_1, \cdots, i_{K-1}]$ be any vector of integers satisfying

$$0 \le i_0 < i_1 < \cdots < i_{K-1} \le (N-1) \tag{199}$$

and let $\boldsymbol{I}_o$ be given by

$$\boldsymbol{I}_o = [0, 1, \cdots, \frac{K}{2} - 1, N - \frac{K}{2}, \cdots, N-2, N-1]. \tag{200}$$

Then

$$\mathrm{var}(\boldsymbol{I}_o) \ge \mathrm{var}(\boldsymbol{I}), \tag{201}$$

where we have equality iff $\boldsymbol{I} = \boldsymbol{I}_o$.

Proof: Let $\boldsymbol{I}_o = [i_{o0}, i_{o1}, \cdots, i_{o(K-1)}]$, *i.e.*

$$\begin{aligned} i_{ok} &= k,\ 0 \le k \le \frac{K}{2} - 1, \\ &= (N-K) + k,\ \frac{K}{2} \le k \le K-1. \end{aligned} \tag{202}$$

The mean of all the elements of $\boldsymbol{I}_o$ is $\mu_o = \frac{N-1}{2}$. The variance of a vector does not change if we shift all the elements by the same amount, and therefore, we now deal with two new vectors $\boldsymbol{I}^1$ and $\boldsymbol{I}_o^1$ formed by shifting $\boldsymbol{I}$ and $\boldsymbol{I}_o$ to the left by μ_o. In symbols,

$$\begin{aligned} \boldsymbol{I}^1 &= [x_0, x_1, \cdots, x_{K-1}], \\ \boldsymbol{I}_o^1 &= [x_{o0}, x_{o1}, \cdots, x_{o(K-1)}], \\ x_k &= i_k - \mu_o, \\ x_{ok} &= i_{ok} - \mu_o. \end{aligned} \tag{203}$$

In addition, we can write

$$\boldsymbol{I}^1 = \boldsymbol{I}_o^1 + \boldsymbol{\epsilon}, \tag{204}$$

where $\boldsymbol{\epsilon} = [\epsilon_0, \epsilon_1, \cdots, \epsilon_{K-1}]$ and $\epsilon_i = x_i - x_{oi}$. The set $\boldsymbol{\epsilon}$ satisfies the following constraints.

$$\begin{aligned} 0 \leq \epsilon_0 \leq \epsilon_1 \leq \cdots \leq \epsilon_{\frac{K}{2}-1} \leq (N-K) \\ -(N-K) \leq \epsilon_{\frac{K}{2}} \leq \epsilon_{\frac{K}{2}+1} \leq \cdots \leq \epsilon_{K-1} \leq 0 \\ \epsilon_{\frac{K}{2}-1} - \epsilon_{\frac{K}{2}} \leq (N-K). \end{aligned} \tag{205}$$

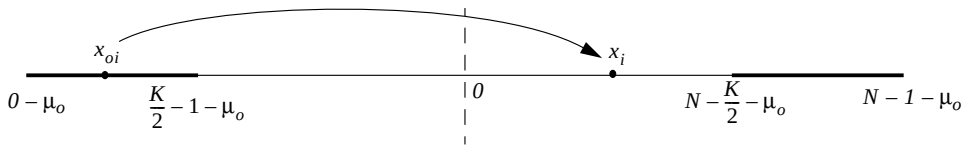

Figure 43: The various constraints of $\boldsymbol{\epsilon}$.

Figure 43 explains the constraints on $\boldsymbol{\epsilon}$. The elements of $\boldsymbol{I}_o^1$ are marked along the real line. The left-half elements are all negative and the only perturbations allowed are positive. The perturbations $\{\epsilon_i\}_0^{\frac{K}{2}-1}$ form an increasing sequence with the maximum being $N-K$, which happens when $i_{\frac{K}{2}-1} = N - \frac{K}{2} - 1$. Similarly, we get the second restriction from the right-half. The third restriction arises from the fact that there are

N slots and K distinct, ordered elements, and therefore, no two consecutive elements can be separated by more than $N-K$.

The variances under consideration, $\mathrm{var}(\boldsymbol{I}_o)$ and $\mathrm{var}(\boldsymbol{I})$, can now be rewritten as

$$\begin{aligned}\mathrm{var}(\boldsymbol{I}_o) &= \frac{1}{K}\sum x_{ok}^2, \\ \mathrm{var}(\boldsymbol{I}) &= \frac{1}{K}\sum x_k^2 - \left(\frac{1}{K}\sum \epsilon_k\right)^2. \end{aligned} \tag{206}$$

Next, consider the left half, i.e., $0 \le k \le \frac{K}{2}-1$. The intuition for the following steps is as follows. For the left half, all the x_{ok}s are negative and positive ϵ_ks will ensure that $|x_k| < |x_{ok}|$ if we make sure that for all possible ϵ_ks, x_k does not exceed $-x_{ok}$. From (202), (203), and (205),

$$0 \le \epsilon_k \le N-K, \tag{207}$$

$$\implies x_{ok} \le x_{ok}+\epsilon_k \le N-K+x_{ok}, \tag{208}$$

$$\implies i_{ok}-\mu_o \le x_k \le i_{ok}-\mu_o+N-K, \tag{209}$$

$$\implies |x_k| \le \max(|i_{ok}-\mu_o|, |i_{ok}-\mu_o+N-K|), \tag{210}$$

$$\implies |x_k| \le \max(\frac{N-1}{2}-k, N-K+\frac{N-1}{2}+k), \tag{211}$$

$$\implies |x_k| \le \frac{N-1}{2}-k, \tag{212}$$

$$\implies |x_k| \le |x_{ok}|. \tag{213}$$

The equality condition is satisfied iff $\epsilon_k = 0$ since $\frac{N-1}{2}-k > N-K+\frac{N-1}{2}+k$ for the left half. By proceeding similarly, we can show that the same holds for the right half. Combining, we get

$$|x_k| \le |x_{ok}| \quad \text{for} \quad 0 \le k \le (K-1), \tag{214}$$

with equality iff $x_k = x_{ok}$ for all k.

Combining (206), (214) and the fact that $\left(\frac{1}{K}\sum \epsilon_k\right)^2 \ge 0$ with equality iff $\boldsymbol{\epsilon} = \mathbf{0}$, we get the desired result, *i.e.*,

$$\mathrm{var}(\boldsymbol{I}_o) \ge \mathrm{var}(\boldsymbol{I}), \tag{215}$$

where we have equality iff $\boldsymbol{I} = \boldsymbol{I}_o$.

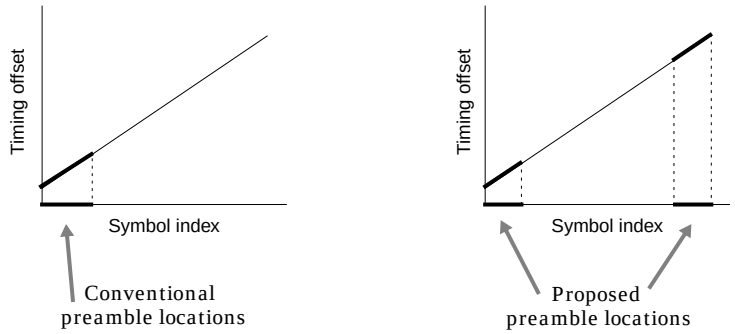

Figure 44: Proposed preamble placement strategy.

Therefore, to minimize $V_{\Delta T}$, we need to split the available preamble slots into two equal halves and place these halves, one at the beginning and one at the end of the packet, as shown in Figure 44.

6.3.3 Dealing with V_{τ_0}

Recall that

$$V_{\tau_0} = \frac{\sigma_n^2}{K}\left(1 + \frac{\text{mean}(\boldsymbol{I})^2}{\text{var}(\boldsymbol{I})}\right). \tag{216}$$

Therefore, the optimal arrangement in this case ($\boldsymbol{I}'$) is

$$\boldsymbol{I}' = \underset{\boldsymbol{I}}{\text{argmax}} \frac{\text{var}(\boldsymbol{I})}{\text{mean}(\boldsymbol{I})^2}. \tag{217}$$

This can be split into two steps: first, we find the optimal arrangement for each possible value of the mean and then find the optimum over all mean values, *i.e.*,

$$\boldsymbol{I}' = \underset{\boldsymbol{I}_k}{\text{argmax}} \frac{\text{var}(\boldsymbol{I}_k)}{\text{mean}(\boldsymbol{I}_k)^2}, \quad \text{where} \quad \boldsymbol{I}_k = \underset{\boldsymbol{I}, \text{mean}(\boldsymbol{I}) = k}{\text{argmax}} \text{var}(\boldsymbol{I}). \tag{218}$$

The solution of this problem leads to the following arrangement. We place K_1 known symbols at the start and $K - K_1$ at the end of the packet where

$$K_1 = \left\lceil K\left(1 - \frac{(3N-K)K}{6N^2}\right)\right\rceil. \tag{219}$$

For $N >> K$, we see that $K_1 \approx K$, which means we need to place almost all the known symbols at the start of the packet to do the best estimation of τ_0. The optimal

arrangement for estimating ΔT performs worse by a factor of approximately 2 as far as the estimation error variance for τ_0 goes. One interpretation of this is that the known symbols at the end of the packet do not contribute at all to the estimation of the initial offset τ_0, and the error variance is roughly equivalent to what we would get if we had only $K/2$ known symbols, all at the beginning.

One way to work around this mismatch of the optimal solutions is to redefine the indices so that the mean for the optimal solution for estimating ΔT is zero, i.e., shift all indices to the left by $\frac{N-1}{2}$ as done in the proof. From (216), we see that having a zero mean is the best possible solution given K symbols. Therefore, the optimal scheme for estimating ΔT performs optimally for τ_0 too. The difference being, in this case, we estimate the timing offset at the center of the packet instead of at one end. Assuming that we are going to use $\hat{\Delta T}$ and $\hat{\tau_0}$ to compute the expected timing offsets for the remaining positions, this arrangement makes intuitive sense as this minimizes the maximum distance over which we need to interpolate, thus minimizing the maximum absolute value of the interpolation error.

6.3.4 Performance Evaluation of the Optimal Placement

In the previous section, it was proved that the optimal preamble placement was the one where we place half the known symbols at the start of the packet and half at the end. We compare three schemes for performance with respect to estimating ΔT:

- all the known symbols at the start of the packet ($\boldsymbol{I}_1$) with error variance V_1;
- the optimal arrangement ($\boldsymbol{I}_2$) with V_2;
- sprinkle the known symbols uniformly throughout the packet ($\boldsymbol{I}_3$) with V_3.

The error variance of the least squares estimator for these three schemes can be evaluated using (198), and they are

$$C_1 \;=\; \frac{V_1}{\sigma_n^2} = \frac{12}{K(K^2-1)},$$

$$
\begin{aligned}
C_2 &= \frac{V_2}{\sigma_n^2} = \frac{1}{\frac{K(K^2-1)}{12} + \frac{KN(N-K)}{4}}, \\
C_3 &= \frac{V_3}{\sigma_n^2} = \frac{(K-1)^2}{(N-1)^2}\frac{V_1}{\sigma^2},
\end{aligned} \tag{220}
$$

where we consider the normalized error variance coefficient C defined as the estimation error variance divided by the observation noise variance.

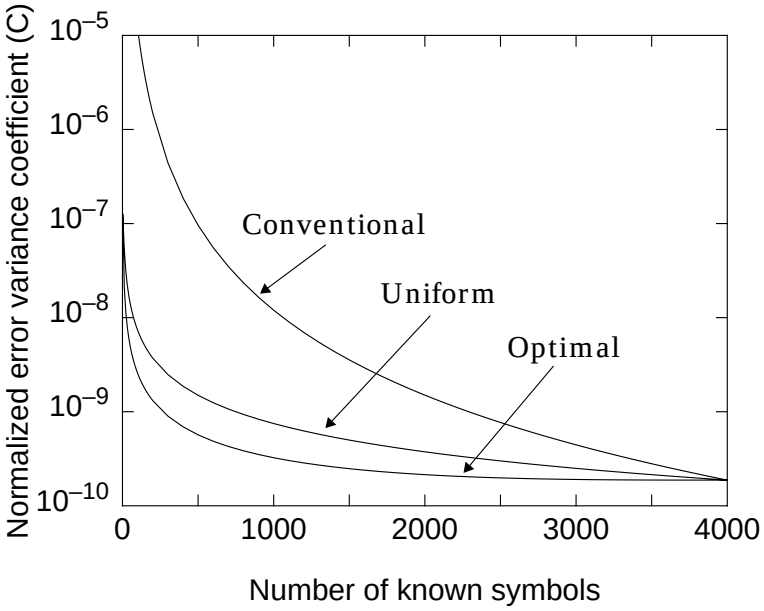

Figure 45: Optimal arrangement much better than conventional one.

Figure 45 compares the performance of the three acquisition schemes for $N = 4000$, and K ranging from 1 to 4000. The optimal arrangement significantly outperforms the conventional one, and the uniform sprinkling scheme is not much worse off. As an example, we compare the total number of symbols needed to get the normalized error variance coefficient $C = 2 \times 10^{-9}$. The optimal scheme needs 86 symbols, the uniform scheme needs 248 symbols, whereas the conventional scheme needs 1588 symbols of the 4000 to be known!

6.4 Actual System: Optimal Preamble Placement

In the previous section, we considered the linearized system and arrived at the best preamble placement strategy for this system. This linear system inherently assumes that we have a PLL operating on the received waveform, where we add the outputs of the TED and the PLL to arrive at the linear measurements. In this section, we use the CRB to show that the result developed in Section 6.3 for the optimal preamble placement applies to the general non-linear problem as well.

Consider once again the general system model:

$$r(t) = \sum_{l=0}^{N-1} a_l h(t - lT - \tau_l) + n_1(t), \tag{221}$$

where T is the bit period, $a_l \in \{\pm 1\}$ are the N *i.i.d.* data symbols, $h(t)$ is the channel impulse response, $n_1(t)$ is additive Gaussian noise band-limited to $[-\frac{1}{2T}, \frac{1}{2T})$, and τ_l is the unknown timing offset for the l^{th} symbol. During acquisition, we are interested only in the initial timing offset and the frequency offset, and therefore, the timing model is $\tau_k = \tau_0 + k\Delta T$. With $h(t)$ band-limited such that uniform baud-rate samples are sufficient statistics to reconstruct $r(t)$, we only store uniform samples r_k given by

$$r_k = \sum_{l=0}^{N-1} a_l h(kT - lT - \tau_0 - l\Delta T) + n_k, \tag{222}$$

where $\{n_k\}$ are *i.i.d.* zero-mean Gaussian random variables of variance σ^2. During acquisition, only K of the N data symbols are known at the receiver and only the observations corresponding to these K positions are used. Assuming that the known symbols can be arbitrarily inserted in the N positions, the CRB on estimating the timing parameters is given by

$$\begin{aligned} V_{\Delta T} &= \frac{\sigma^2}{KE_{h'}} \frac{1}{\frac{1}{K}\sum x_m^2 - (\frac{1}{K}\sum x_m)^2} &= \frac{\sigma^2}{KE_{h'}} \frac{1}{\mathrm{var}(\boldsymbol{I})}, \\ V_{\tau_0} &= \frac{\sigma^2}{KE_{h'}} \frac{\frac{1}{K}\sum x_m^2}{\frac{1}{K}\sum x_m^2 - (\frac{1}{K}\sum x_m)^2} &= \frac{\sigma^2}{KE_{h'}} \left(1 + \frac{\mathrm{mean}(\boldsymbol{I})^2}{\mathrm{var}(\boldsymbol{I})}\right), \end{aligned} \tag{223}$$

where $\boldsymbol{I}$ is the vector of acquisition indices $\{x_k\}$ and $E_{h'}$ is the energy in the derivative of the channel impulse response $h(t)$. This has the same structure as that with the

linear model after the operation of the PLL. Therefore, for the actual non-linear system, the optimal preamble placement to minimize the CRB is the same one, where we split the available known symbols into two halves and place them at the beginning and at the end of the packet. For the frequency offset model, the CRB can be achieved using the Levenberg-Marquardt method (Chapter 3) and therefore, the arrangement that minimizes the CRB indeed gives the best performance. In summary, whether we use the PLL or a CRB-achieving timing recovery scheme, the best preamble placement strategy is the same.

6.5 Cycle Slips

An added advantage with the optimal scheme is that it is well equipped to handle cycle-slips. We can use independent estimators at the beginning and at the end of the packet to locate the start and the end preamble sequences, thus reducing the effects of cycle slips in the tracking mode in between.

We present simulation results for the following simple acquisition strategy. The known symbols are placed half at the beginning and half at the end of the packet. Let the two preamble sequences be denoted by S_1 (beginning) and S_2 (end) respectively. At the receiver, we perform correlation detection and run a trained PLL, trained to the sequence S_1. At the end of the first preamble sequence, the PLL is switched to the decision-directed mode and operated in that mode till the end of the packet. Next, out of the N decisions available, we pick the last $K/2$ and correlate these with the $K/2$ symbols corresponding to S_2 for different shifts and perform peak detection. This helps in detecting cycle slips. This knowledge is then used to suitably correct the timing estimates $\hat{\tau}_k$ corresponding to the location of S_2 in the received sequence.

Figure 46 shows the improvement due to the aforementioned acquisition scheme when compared to the conventional scheme where all the known symbols are placed at the beginning of the packet. We consider the uncoded PR-IV system with $N = 5000$

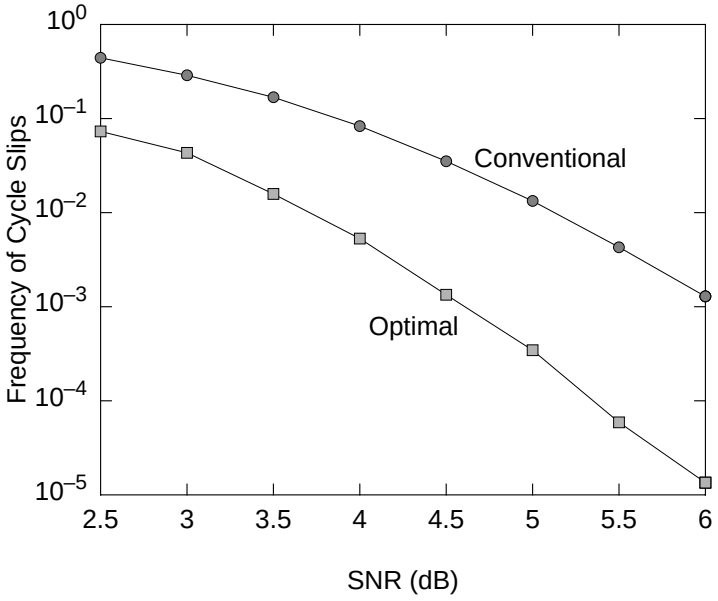

Figure 46: Splitting the preamble reduces the occurrence of cycle slips.

being the packet size. 120 of these symbols are known at the receiver. For the conventional system, we use the length-120 periodic version of the 1100 pattern. For the proposed scheme, we have a length-60 sequence formed by repetitions of the 1100 pattern for the preamble sequence at the start of the packet. This periodic pattern helps in rapid acquisition of the frequency offset. For the known sequence at the end of the packet, we choose a random length-60 sequence. This is to get good correlation performance. The PLL gain parameters are $\alpha = 0.04$ and $\beta = \alpha^2/4$, chosen to minimize the number of cycle slips for the conventional system at an SNR of 5 dB. For the optimal arrangement, the number of cycle slips that were not detected and/or corrected is plotted. The acquisition scheme described above reduced the number of residual cycle slips by a factor of 10 to 100 depending on the SNR. In terms of SNR, at a cycle slip rate of 10^{-3}, we observe an improvement of around 1.5 dB.

We also simulated the system where all the data symbols were known at the receiver, *i.e.*, the trained system. The trained system did not exhibit any cycle slips for the range of SNRs considered when the PLL-based system was used for timing

recovery. Again, this leads us to the conclusion that one way to improve the timing recovery performance is to improve the quality of the decisions available to the timing recovery block.

6.6 Joint Acquisition and Tracking

So far we have considered the tracking and acquisition problems in isolation. Here we consider a simple version of the problem of joint acquisition and tracking. We ignore the presence of the random walk, and assume that the timing offset model consists only of an initial timing offset τ_0 and a frequency offset parameter ΔT.

When we considered a system where we performed only acquisition, we assumed the following linear observation model.

$$\boldsymbol{y} = \boldsymbol{\tau} + \boldsymbol{\nu}, \tag{224}$$

where $\boldsymbol{y} = [y_0\ y_1\ \ldots\ y_{K-1}]^T$, $\boldsymbol{\tau} = [\tau_0\ \tau_1\ \ldots\ \tau_{K-1}]^T$, and $\boldsymbol{\nu} = [\nu_0\ \nu_1\ \ldots\ \nu_{K-1}]^T$. The noise variables $\{\nu_k\}$ were assumed to be *i.i.d.* zero-mean normal random variables of variance σ_n^2. In this section, we postulate the existence of a timing recovery block that takes advantage of the K known symbols in the training phase, leading to a measurement noise variance σ_a^2 for the observations corresponding to the known symbols (acquisition), and σ_t^2 for the observations corresponding to the data symbols (tracking) such that $\sigma_a^2 < \sigma_t^2$. An example is the PLL-based timing recovery system, where the timing estimation error variance is lesser with training than otherwise. Though the timing estimation error variance as a function of the bit position does not show the step-function type behavior assumed here, but rather varies in a gradual fashion, we use this model as a close approximation.

Let the parameter to be estimated be $\boldsymbol{\theta} = [\Delta T\ \tau_0]^T$, and let $\boldsymbol{x}(\boldsymbol{\theta})$ be the noiseless observation. For the linear model of (224), the k^{th} component of $\boldsymbol{x}(\boldsymbol{\theta})$ is just $x_k = \tau_k = k\Delta T + \tau_0$. The noise variables are not *i.i.d.* Therefore, to evaluate the Fisher

information matrix, we use (46), which is reproduced here:

$$\boldsymbol{J}_\theta = \boldsymbol{G}^T \boldsymbol{R}^{-1} \boldsymbol{G}, \tag{225}$$

where $\boldsymbol{G}$ is the *sensitivity matrix*

$$\boldsymbol{G} = \left[\frac{\partial}{\partial \boldsymbol{\theta}} \boldsymbol{x}^T(\boldsymbol{\theta}) \right] = \ [\boldsymbol{g_0}\ \boldsymbol{g_1} \cdots \boldsymbol{g_{P-1}}], \tag{226}$$

P is the size of the parameter vector $\boldsymbol{\theta}$ and the $N \times 1$ column vector $\boldsymbol{g_i} = \frac{\partial \boldsymbol{x}(\boldsymbol{\theta})}{\partial \theta_i}$ denotes the variation in the mean $\boldsymbol{x}(\boldsymbol{\theta})$ with respect to θ_i, the i^{th} element of $\boldsymbol{\theta}$. The CRB is given by the inverse of the Fisher information matrix. Applying this to our model, we get

$$\boldsymbol{J}_\theta = \begin{bmatrix} \frac{1}{\sigma_a^2} \sum_{k \in \boldsymbol{I}_a} k^2 + \frac{1}{\sigma_t^2} \sum_{k \in \boldsymbol{I}_t} k^2 & \frac{1}{\sigma_a^2} \sum_{k \in \boldsymbol{I}_a} k + \frac{1}{\sigma_t^2} \sum_{k \in \boldsymbol{I}_t} k \\ \frac{1}{\sigma_a^2} \sum_{k \in \boldsymbol{I}_a} k + \frac{1}{\sigma_t^2} \sum_{k \in \boldsymbol{I}_t} k & \frac{1}{\sigma_a^2} K + \frac{1}{\sigma_t^2}(N-K) \end{bmatrix}, \tag{227}$$

where the ordered vectors $\boldsymbol{I}_a$ and $\boldsymbol{I}_t$ contain the indices corresponding to the training and the data symbols respectively. The elements of $\boldsymbol{I}_a$ and $\boldsymbol{I}_t$ span all the integers from 0 to $N-1$. Let m_a and m_t be the mean of $\boldsymbol{I}_a$ and $\boldsymbol{I}_t$ respectively, and let their respective variances be V_a and V_t. The determinant of $\boldsymbol{J}_\theta$ can be written as

$$|\boldsymbol{J}_\theta| = \left\{ \frac{K}{\sigma_a^2} + \frac{N-K}{\sigma_t^2} \right\} \left\{ \frac{K}{\sigma_a^2} V_a + \frac{N-K}{\sigma_t^2} V_t \right\} + \frac{K}{\sigma_a^2} \frac{N-K}{\sigma_t^2} (m_a - m_t)^2. \tag{228}$$

This determinant can be thought of as a weighted variance of all possible indices, with the weights being $1/\sigma_a^2$ or $1/\sigma_t^2$ depending on whether the indices belong to the acquisition or the tracking sets. The CRB then evaluates to

$$\begin{aligned} V_{\Delta T} &= \frac{1}{|\boldsymbol{J}_\theta|} \left[\frac{K}{\sigma_a^2} + \frac{N-K}{\sigma_t^2} \right], \\ V_{\tau_0} &= \frac{1}{|\boldsymbol{J}_\theta|} \left[\frac{K}{\sigma_a^2}(V_a + m_a^2) + \frac{N-K}{\sigma_t^2}(V_t + m_t^2) \right]. \end{aligned} \tag{229}$$

When performing acquisition alone, to minimize $V_{\Delta T}$, we needed to maximize the variance of the set of acquisition indices. To minimize $V_{\Delta T}$ with joint acquisition and tracking, from (229), we see that we need to pick the acquisition indices to maximize

the determinant $|\boldsymbol{J}_\theta|$. As before, we shift the indices so that their possible indices are of the form $i-(N-1)/2$ for $i=0$ to $N-1$, and these are apportioned into the vectors $\boldsymbol{I}'_a$ and $\boldsymbol{I}'_t$ with mean and variance being m'_a, V'_a and m'_t, V'_t respectively. Shifting by a constant does not change the variance, and therefore, $V'_a = V_a$ and $V'_t = V_t$. Also, we get $m'_a - m'_t = m_a - m_t$. So, from (228), we see that this shift does not affect the evaluation of the determinant.

Now, consider the optimal arrangement of Section 6.3.2, where the acquisition positions are chosen assuming we perform acquisition alone. Let this be set of acquisition indices when we perform acquisition and tracking jointly. Therefore,

$$\boldsymbol{I}_a^o = [0, 1, \cdots, \frac{K}{2} - 1, N - \frac{K}{2}, \cdots, N-2, N-1]. \tag{230}$$

For this set-up, the means of the shifted vectors m'_a and m'_t are zero and the determinant can be rewritten as

$$\begin{aligned} |\boldsymbol{J}_\theta|^o &= \left\{\frac{K}{\sigma_a^2} + \frac{N-K}{\sigma_t^2}\right\}\left\{\frac{1}{\sigma_a^2}\sum_{k\in\boldsymbol{I'}^o_a} k^2 + \frac{1}{\sigma_t^2}\sum_{k\in\boldsymbol{I'}^o_t} k^2\right\}, \\ &= \left\{\frac{K}{\sigma_a^2} + \frac{N-K}{\sigma_t^2}\right\}\left\{\frac{1}{\sigma_t^2}\sum_{k\in\boldsymbol{I'}} k^2 + \left(\frac{1}{\sigma_a^2} - \frac{1}{\sigma_t^2}\right)\sum_{k\in\boldsymbol{I'}^o_a} k^2\right\}, \\ &= C_1\left(C_2 + C_3\sum_{k\in\boldsymbol{I'}^o_a} k^2\right), \end{aligned} \tag{231}$$

where $\boldsymbol{I'}$ is the ordered vector containing all shifted indices $i-(N-1)/2$ for $i=0$ to $i=N-1$, which is a constant, independent of the choice of $\boldsymbol{I}_a$. The definition of the constants C_1, C_2 and C_3 follow from equations above. $|\boldsymbol{J}_\theta|^o$ is a function of only the variance of $\boldsymbol{I'}^o_a$.

For an arbitrary choice $\boldsymbol{I}_a$, the determinant evaluates to

$$|\boldsymbol{J}_\theta| = C_1\left(C_2 + C_3\sum_{k\in\boldsymbol{I'}_a} k^2\right) - C_3^2\left(\sum_{k\in\boldsymbol{I'}_a} k\right)^2. \tag{232}$$

It was shown in Section 6.3.2 that the choice $\boldsymbol{I}_a = \boldsymbol{I}_a^o$ maximizes the term $\sum_{k\in\boldsymbol{I'}_a} k^2$. In addition, $\sum_{k\in\boldsymbol{I'}^o_a} k = 0$. Therefore, the choice $\boldsymbol{I}_a = \boldsymbol{I}_a^o$ maximizes

$|\boldsymbol{J}_\theta|$, leading to a maximum $|\boldsymbol{J}_\theta| = |\boldsymbol{J}_\theta|^o$. We considered only $V_{\Delta T}$ here. As with the acquisition case, it can also be shown that the same arrangement also minimizes V_{τ_0}, with the caveat that now we estimate the intercept value at the center of the packet as opposed to the value at the start of the packet. In summary, for the system model considered here, the choice that minimizes the CRB for acquisition alone minimizes the CRB for the joint acquisition and tracking case as well.

6.7 Summary

In this chapter, we studied the problem of acquisition, where the goals are to accurately estimate the initial phase offset and the frequency offset so that the tracking mechanism functions properly. The conventional acquisition method is to place known symbols at the start of the packet and use these to operate the PLL-based tracking system in the trained mode. If all the known symbols can be placed arbitrarily throughout the packet, we show that to minimize the CRB on the timing estimation error with a frequency offset model, we need to place the known symbols half at the beginning of the packet and half at the end. This arrangement also reduces the occurrence of cycle slips greatly. In addition, in the absence of a random walk during the tracking mode, we showed that the same arrangement also minimizes the CRB for joint acquisition and tracking.

CHAPTER 7

ITERATIVE TIMING RECOVERY

In Chapters 3 to 6, we studied timing recovery methods for the uncoded system, and in Chapter 2, we described the coded system. In this chapter, we propose iterative timing recovery for the coded system and present simulation results to demonstrate its performance.

In a conventional receiver, timing recovery is performed assuming an uncoded system. This approach works well because the SNR of operation is high enough that the conventional PLL-based timing recovery methods work well enough. The samples generated based on the timing recovery block's estimates are then processed by the equalizer and the decoder to get the final decisions.

With the advent of iteratively decodable codes [9] [25], the SNR of operation could be lowered significantly. Lower SNR reduces the cost of operations, and in magnetic recording systems, allows for higher data densities, and therefore, is a very desirable property. In addition, turbo equalization [54] allowed for the use of simpler ECCs and led to a further reduction in the operating SNRs. In this low SNR regime, the conventional PLL-based timing recovery schemes are tested severely.

The optimal solution requires the joint estimation of the timing recovery offsets and the data bits. In effect, we need to perform the tasks of timing recovery, equalization and ECC decoding jointly. Such joint detection may be performed using the expectation-maximization (EM) algorithm [19] [6]. However, a solution based on the EM algorithm [26] turns out to be computationally prohibitive.

We propose iterative timing recovery as an alternative to the conventional approach that separates timing recovery and turbo equalization. Soft information from

the turbo equalizer iterations is fed back to the timing recovery block which uses this information to refine the samples being fed to the turbo equalizer for further iterations. Thus, as iterations progress, the performance of both the timing recovery block and the turbo equalizer improves. With the PLL-based system as the timing recovery block, we observe significant reduction in the SNR required at the expense of a small increase in complexity. Using sophisticated timing recovery algorithms instead of the PLL-based one leads to marginal improvements, supporting the conjecture that, in an iterative setting, it is the process of iterations itself that buys us the improvements as opposed to optimizing the individual components.

Next, we consider the effect of the optimal acquisition strategy developed in Chapter 6. Using this reduces the occurrence of cycle slips greatly and this, in turn, reduces the complexity and improves the performance of the system. Finally, we compare the complexity of the different timing recovery methods considered here.

The rest of this chapter is organized as follows. In Section 7.1, we briefly summarize the conventional system that separates timing recovery and turbo equalization. In Section 7.2, we present simulation results for two systems, and motivate the iterative timing recovery algorithm. In Section 7.3, we propose iterative timing recovery. This method is simulated for a variety of systems and analyzed in Section 7.4. Finally, the results and observations are summarized in Section 7.5.

7.1 Conventional System

In a conventional receiver, the continuous-time received waveform is first sampled using the PLL-based timing recovery architecture described in Chapter 4. Then, the samples are acted on by the turbo equalizer described in Chapter 2 to produce the receiver's estimates of the transmitted message bits.

We consider two types of error control codes (ECCs) in this chapter: convolutional codes and LDPC codes. The block diagrams corresponding to these two cases are

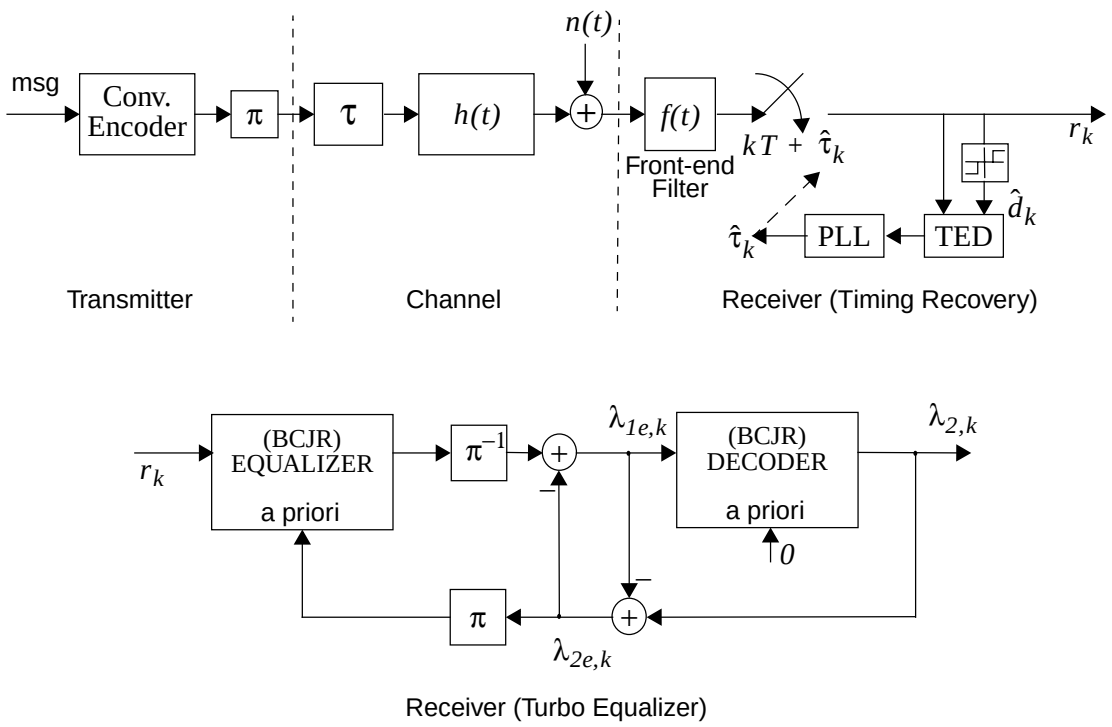

Figure 47: Block diagram of a conventional convolutional coded system.

slightly different. The system block diagram for the case with the convolutional code as the ECC is shown in Figure 47. In the case of LDPC codes, the interleaver after the ECC and before the channel is redundant since LDPC codes are random codes. The action of the interleaver is equivalent to relabeling the bit nodes of the graph, and this does not affect the properties or the performance of the code. The block diagram in the case of the LDPC code as the ECC is shown in Figure 48.

7.2 Motivation for Iterative Timing Recovery

The conventional method of separating timing recovery and turbo equalization performs poorly when faced with even moderate timing offsets. Consider first the following system: The outer code is a rate-1/4 convolutional encoder with generator polynomial given by $[1, 1, 1, \frac{1 \oplus D}{1 \oplus D \oplus D^2}]$, which maps blocks of 1278 message bits onto blocks of 5120 coded bits. The channel is the precoded PR-IV channel characterized

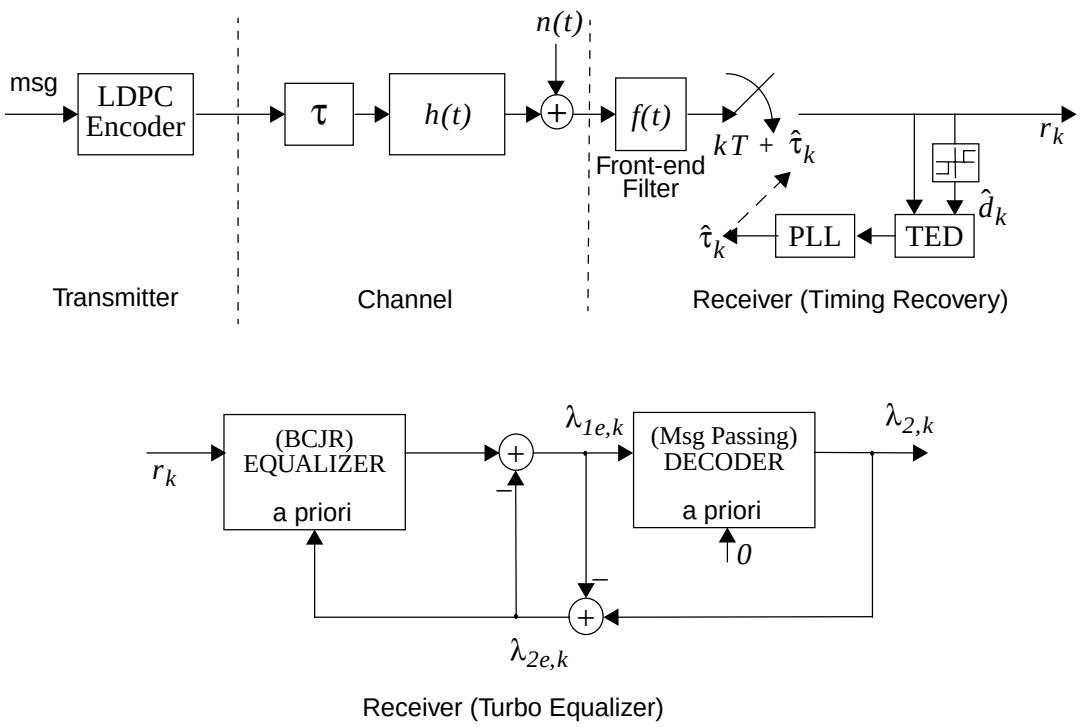

Figure 48: Block diagram of a conventional LDPC coded system.

by the rate-1 generator polynomial $[\frac{1-D^2}{1\oplus D^2}]$. The rate of this serial turbo code is low, and consequently, the SNR of operation of the timing recovery block is low. We consider a random walk timing offset with $\sigma_w/T = 0.3\%$. This represents a moderate random walk with low probability of cycle slips. We assume perfect acquisition. Since we do not have a frequency offset, a first order PLL is sufficient, and the gain factor is $\alpha = 0.025$, chosen to minimize the mean squared timing estimation error at SNR = 5 dB. The interleaver is an s-random interleaver with $s = 16$, and at most 50000 packets were used for each SNR. The bit-error rate (BER) vs. SNR curves for this system are shown in Figure 49. Also plotted are the performance curves with known timing and for the genie-aided system. For the known timing system, the timing offsets are all known at the receiver, and we sample at the right sampling instants. In the case of the genie-aided system, the timing recovery block has access to the transmitted symbols, and therefore, operates in the trained mode. Both these give

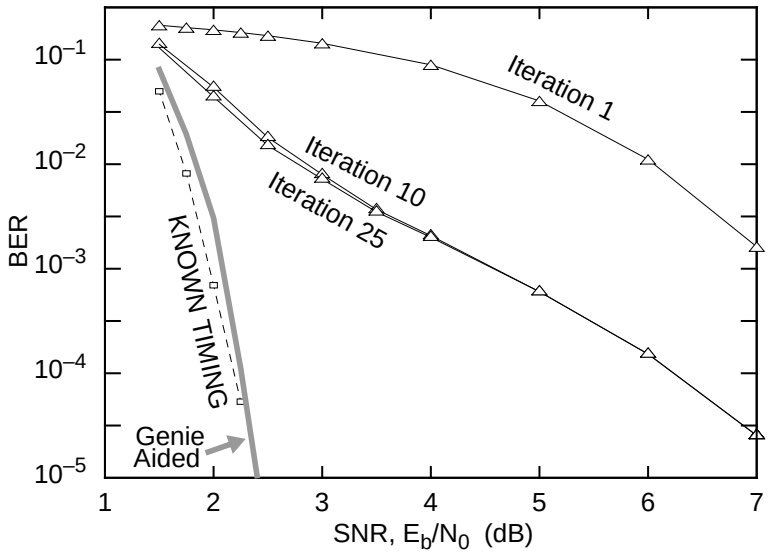

Figure 49: The rate-1/4 convolutional coded system is more than 4.5 dB away from known timing.

lower bounds to the performance of the conventional system. After 25 iterations of the turbo equalizer, the conventional scheme is more than 4.5 dB away from known timing at BER $= 10^{-4}$, and the genie-aided system is less than 0.2 dB away from the known timing case. The loss of performance for the conventional system is largely due to the unreliability of the decisions fed to the timing recovery device, and with right decisions, *i.e.*, in the genie-aided case, the loss is less than 0.2 dB.

The next example is a coded system with a high-rate LDPC outer code. Specifically, we consider a rate-8/9 $(3, 27)$-regular LDPC code of coded block length 4095. The parity-check matrix for this code has 3 ones in each column and 27 ones in each row. We again assume the precoded PR-IV channel characterized by the rate-1 generator polynomial $[\frac{1-D^2}{1 \oplus D^2}]$. The timing offset is a pure frequency offset with the parameter $\Delta T = 0.2\%$ and perfect acquisition implying $\tau_0 = 0$. We use a second order PLL to track the frequency offset with the gain parameters being $\alpha = 0.04$, chosen to minimize the timing estimation error variance at SNR $= 5$ dB, and $\beta = \alpha^2/4$. A maximum of 3×10^6 packets were simulated. The sector-error rate (SER) for this

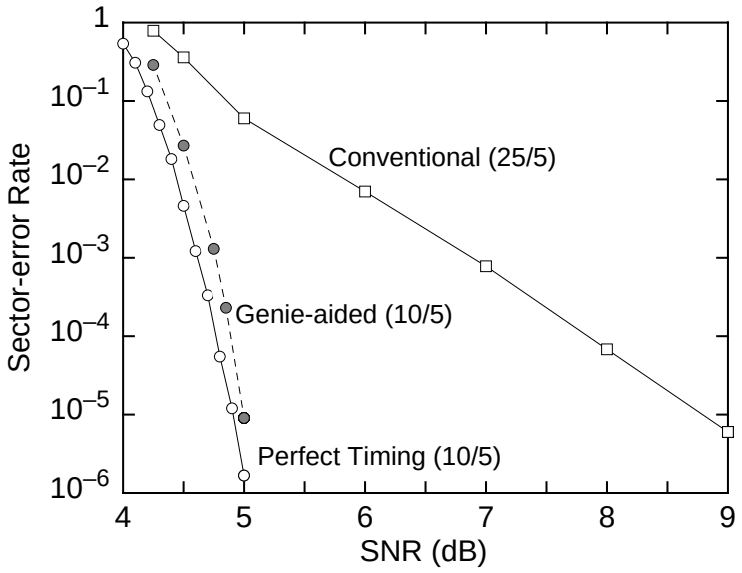

Figure 50: The rate-8/9 LDPC coded system is around 4 dB away from known timing.

system is plotted as a function of SNR in Figure 50. The notation for the iterations is x/y where x is the total number of turbo iterations, and y is the number of message passing iterations performed for each call of the LDPC decoder. Again, in this case, we observe the same general conclusions. At SER $= 10^{-5}$, the conventional system is about 4 dB away from the perfect timing case, and the genie-aided system system is less than 0.2 dB away from the perfect timing case.

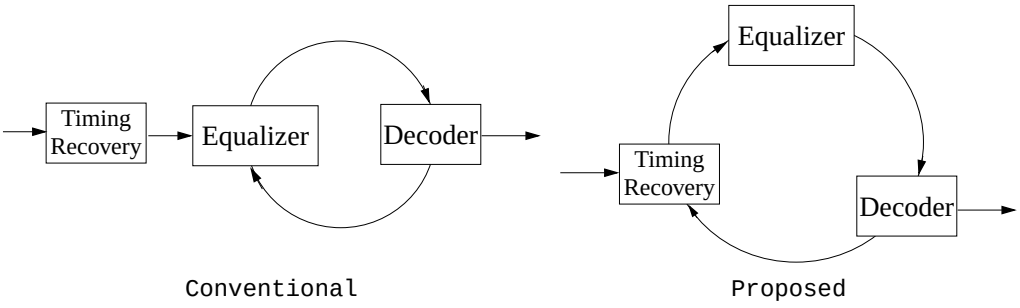

Figure 51: Comparing the conventional and the proposed receivers.

Both these examples provide strong motivation for iterative timing recovery, where we pass the better decisions available at the end of turbo iterations back to the

timing recovery device, thus improving its performance. The difference between the conventional and the proposed receivers is summarized in Figure 51. The timing recovery block in the proposed receiver exploits the presence of the turbo equalizer, whereas the conventional receiver does not. The proposed receiver is described in more detail in the sequel.

7.3 Iterative Timing Recovery

Consider a coded system where the outer code is a convolutional code. Consider a conventional receiver with PLL-based timing recovery followed by a turbo equalizer. After the first iteration, the turbo equalizer produces soft symbol estimates $\{\tilde{d}_k\}$ that would be more reliable than the tentative decisions of the memoryless soft slicer used by the PLL. If we were to run the front-end PLL again using the original read-back waveform but using the soft decisions from the turbo equalizer, we would get an improved set of timing estimates $\{\hat{\tau}_k^{new}\}$. Rather than store the continuous-time readback waveform in its entirety, we would only need to store the original set of samples, since the band-limited nature of $r(t)$ makes them sufficient statistics. Thus, the second pass of the PLL could arrive at new samples $\{r_k^{new}\}$ through an interpolation of $\{r_k\}$, according to:

$$r_k^{new} = \sum_l r_l p(kT - lT + \hat{\tau}_k^{new} - \hat{\tau}_l), \tag{233}$$

where $p(t) = \frac{\sin \pi t/T}{\pi t/T}$.

We now describe the proposed receiver, which is shown in Figure 52. It begins as described above for the first iteration, with a real-time PLL feeding samples $\{r_k\}$ to a turbo equalizer, which feeds soft estimates $\{\tilde{d}_k\}$ to a second PLL which produces improved timing estimates $\{\hat{\tau}_k^{new}\}$. The read-back waveform is then effectively resampled at the improved sampling instants using interpolation of the original samples as described in (233). These new samples are then used in the second iteration of the turbo equalizer. The process then repeats: after each iteration of the turbo equalizer,

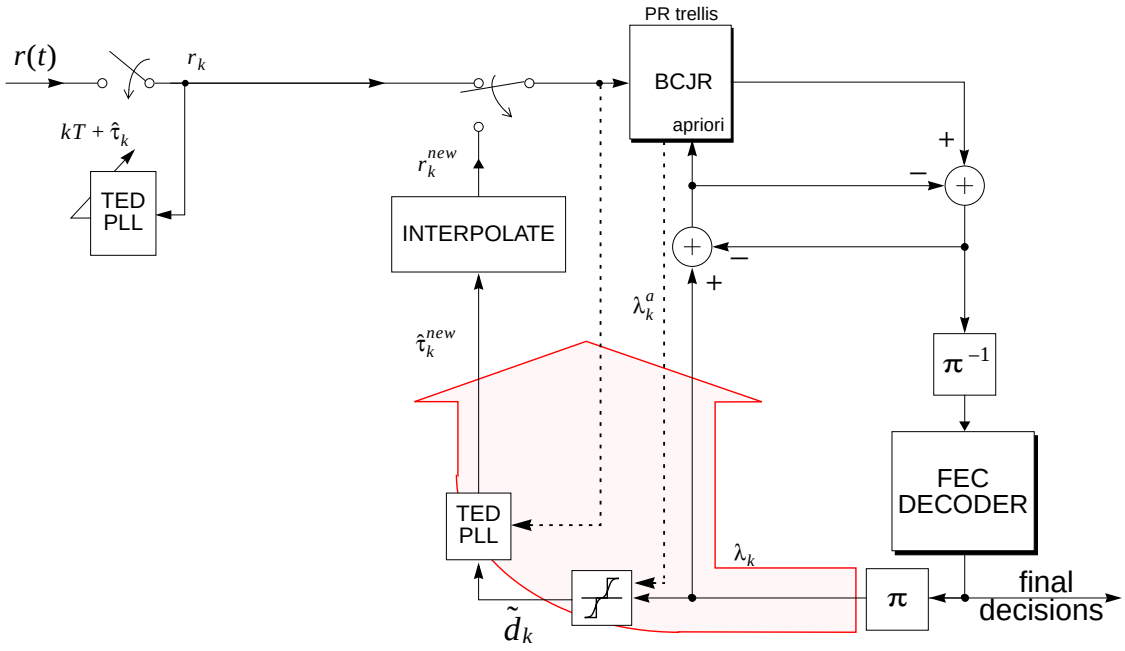

Figure 52: Joint timing recovery and turbo equalization.

soft estimates from the turbo equalizer are used to improve the timing estimates, which are then used to interpolate the original samples before going on to the next turbo iteration.

The proposed receiver of Figure 52 is essentially a modified turbo equalizer, with an interpolation step inserted between consecutive iterations. The complexity increase is marginal, because the complexity of interpolation is usually negligible relative to each turbo iteration. It is worth noting that although we perform timing recovery and turbo equalization jointly, the front-end has remained unchanged, and we still sample the continuous time waveform only once. The modified turbo equalizer is able to correct for poor timing at the front-end PLL.

The block diagram of Figure 52 is for a convolutional outer code. With an LDPC outer code, the block diagram is the same except for the interleaver and the deinterleaver.

7.4 Simulation Results

In this section, we present simulation results to demonstrate the performance of iterative timing recovery with a variety of channel codes and also with different timing offset models. Specifically, we consider the following cases:

- Convolutional outer code:
  - Rate-1/4 code with random walk and PR channel,
  - Rate-8/9 code with random walk and PR channel,
- LDPC outer code:
  - Rate-1/2 irregular code with constant timing offset and AWGN channel,
  - Rate-8/9 irregular code with random walk and PR channel,
  - Rate-8/9 regular LDPC code with frequency offset and PR channel.

7.4.1 Low Rate Convolutional Code + Moderate Random Walk

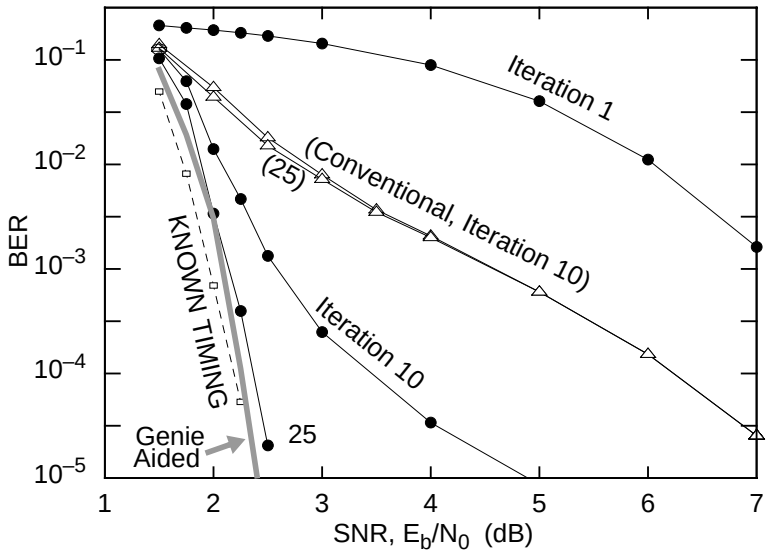

Figure 53: Rate-1/4 convolutional code, random walk, PR channel.

We consider the system whose results with a conventional receiver are shown in Figure 49. The outer code is a rate-1/4 convolutional encoder with generator

polynomial given by $[1, 1, 1, \frac{1\oplus D}{1\oplus D\oplus D^2}]$, which maps blocks of 1278 message bits onto blocks of 5120 coded bits. The channel is the precoded PR-IV channel characterized by the rate-1 generator polynomial $[\frac{1-D^2}{1\oplus D^2}]$. We consider a random walk timing offset with $\sigma_w/T = 0.3\%$, and perfect acquisition. The first-order PLL gain factor is $\alpha = 0.025$, chosen to minimize the mean squared timing estimation error at SNR = 5 dB. The interleaver is an s-random interleaver with $s = 16$, and at most 50000 packets were used for each SNR.

One minor complication with the precoded PR-IV system is that the second PLL requires soft estimates of the precoder output $\{\tilde{d}_k\}$, but a conventional turbo equalizer for the precoded PR-IV channel produces LLRs $\{\lambda_k\}$ for the precoder input $\{b_k\}$. Fortunately it is a simple matter to augment the SISO equalizer so that it produces both $\{\tilde{d}_k\}$ and $\{\lambda_k\}$. Specifically, the BCJR can track LLRs $\{\lambda_k^{'}\}$ for $\{a_k\}$, and use

$$\tilde{d}_k = -2\tilde{a}_{k-2}/(1 + e^{-\lambda_k}), \tag{234}$$

where $\tilde{a}_k = \tanh(\lambda_k^{'}/2)$.

The bit-error rate (BER) vs. SNR curves for this configuration is shown in Figure 53. At BER $= 2 \times 10^{-5}$ and after 25 iterations, we observe a performance gain of 4.7 dB over a conventional system with separate timing recovery and turbo equalization. The performance of the proposed system is 0.2 dB from a turbo equalizer with perfect timing ($\hat{\tau}_k = \tau_k$).

7.4.2 High Rate Convolutional Code + Severe Random Walk

Next, we consider a rate-8/9 system in which blocks of 3636 bits are encoded by the rate-1/2 generator $[1, \frac{1\oplus D\oplus D^3\oplus D^4}{1\oplus D\oplus D^4}]$, and then punctured to a block length of 4095 bits by retaining only every eighth parity bit. As before, the channel is precoded PR-IV and we assume additive white Gaussian noise. To test the performance in the face of cycle slips, we increase the severity of the random walk jitter to $\sigma_w/T = 0.7\%$, which was found to increase the occurrence of cycle slips.

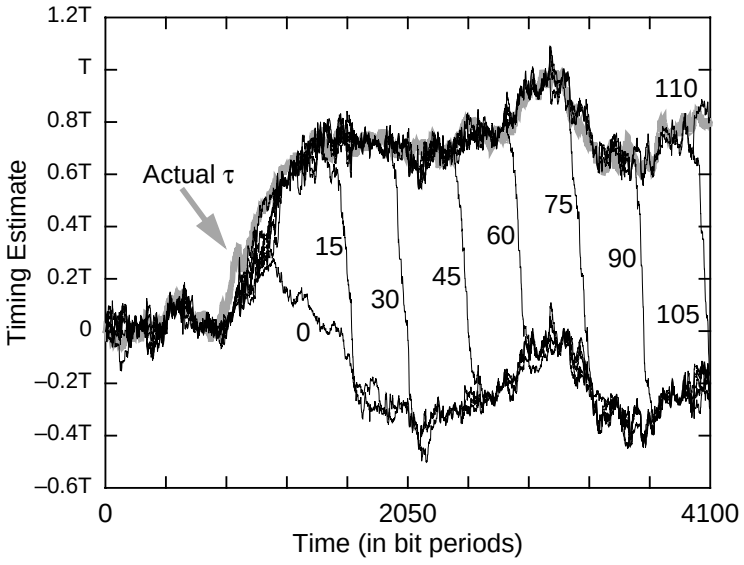

Figure 54: Iterative timing recovery corrects cycle slips automatically.

A benefit of the proposed receiver is that it can correct cycle slips. Figure 54 shows the timing waveforms for a sample packet for the rate-8/9 system at SNR = 5.0 dB and $\sigma_w/T = 0.7\%$. The gray curve represents the actual τ sequence, the curve labeled 0 shows $\hat{\tau}$ after the first pass of the PLL, and the other curves show $\hat{\tau}$ after the number of iterations indicated by the corresponding label. We see that the first-pass PLL is not able to track the rapid increase in the actual τ sequence that occurs after about 1000 symbols; instead, $\hat{\tau}$ wanders downward for a few hundred symbol periods until it eventually converges to approximately $\tau - T$, which represents a cycle slip. However, by the time we reach the 15^{th} iteration, the region where the PLL had wandered has been corrected, so that the PLL transitions from perfect lock to a cycle slip in a very short period of time. The resulting steep slope forms a boundary between perfect lock and cycle slip, and this boundary moves from left to right as iterations progress, until eventually the cycle slip is eliminated.

In practice we can reduce the number of required iterations by detecting the cycle slip and correcting for it [30]. Although cycle-slip detection is difficult in general, it is made easy in our iterative receiver, because a cycle slip eventually leads to an abrupt

change in $\hat{\tau}$ by $\pm T$, as shown in Figure 54. Hence, a simple and effective detection method is to declare a slip whenever the magnitude of $\delta_k = \hat{\tau}_k - \hat{\tau}_{k-d}$ exceeds a given threshold H, for some delay d. To correct the slip, the receiver need only add $\pm T$ to all $\hat{\tau}$ after the slip occurs, with the sign determined by the sign of δ_k.

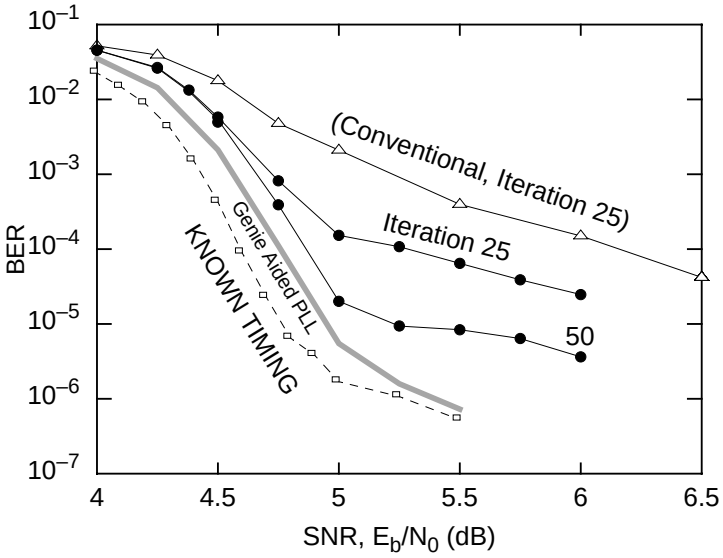

Figure 55: Rate-8/9 convolutional code, severe random walk, PR channel.

Figure 55 shows BER vs. SNR for the proposed system with cycle-slip detection and correction, with $H = 0.75T$, $d = 100$, $s = 24$, $\sigma_w/T = 0.7\%$, $\alpha = 0.04$, chosen as for the rate-1/4 system, and a maximum of 150000 packets per SNR. At SNR = 7 dB, the BER was 2×10^{-5} after 50 iterations for the conventional system (not shown). Therefore, at BER $= 2 \times 10^{-5}$ after 50 iterations, the proposed receiver is 2 dB better than a conventional receiver and 0.3 dB from a turbo equalizer with perfect timing. The genie-aided receiver suffers a 0.2 dB penalty relative to a receiver with perfect timing. This gap can be closed only by discarding the PLL as the basis for translating symbol estimates to timing estimates. The proposed receiver is 0.1 dB worse than the genie-aided receiver, a loss that can be attributed to a nonzero BER after turbo equalization.

7.4.3 Low Rate Irregular LDPC Code + Constant Offset

We next shift attention to LDPC outer codes and begin with rate-1/2 irregular LDPC code characterized by the following edge degree distribution polynomials:

$$\begin{aligned} \lambda_{\text{bit}}(x) &= 0.25105x^2 + 0.30938x^3 + 0.00104x^4 + 0.43853x^{10} \\ \rho_{\text{check}}(x) &= 0.63676x^7 + 0.36324x^8. \end{aligned} \tag{235}$$

We encode blocks of $K = 2500$ bits to get a coded block length of $N = 5000$. The channel is an AWGN channel with no ISI. We have a constant timing offset $\tau = \pi/20$. $\alpha = 0.01$. In effect, this is a simplified system designed to analyze the effects of the outer code on timing recovery.

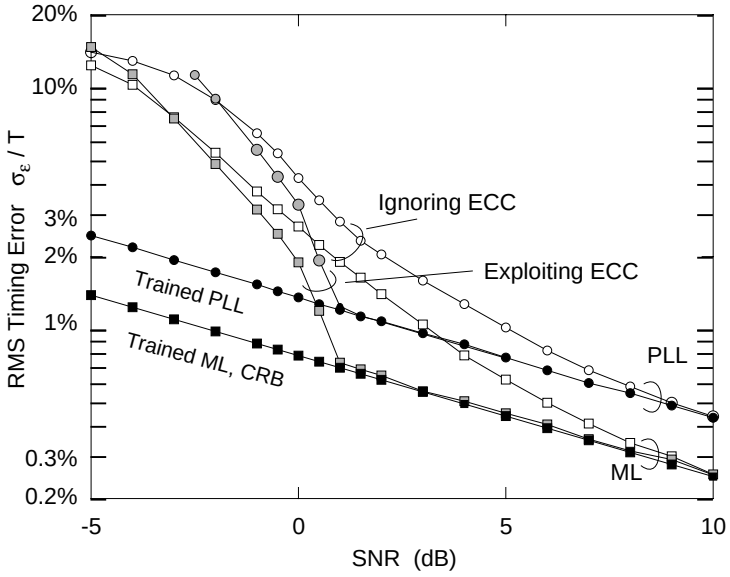

Figure 56: Rate-1/2 irregular LDPC code, constant offset, AWGN channel.

Figure 32 compares the performance of the PLL-based timing recovery, the ML timing recovery and the CRB in the trained and the memoryless soft-decision directed cases. In Figure 56, we show the effect of exploiting the outer code in improving timing recovery by jointly performing timing recovery and decoding. After the water-fall region of the LDPC code, the performance of the iterative receiver matches that

of the trained system for both the ML and the PLL cases. Quantitatively, exploiting the LDPC code reduces the SNR requirement by about 3.3 dB in both the ML and the PLL-based cases.

7.4.4 High Rate Irregular LDPC Code + Random Walk

Next, we consider the rate-8/9 irregular LDPC code characterized by the following node degree distribution polynomials:

$$\begin{aligned} \lambda_{\text{bit}}(x) &= 0.38767x^2 + 0.39823x^3 + 0.14688x^6 + 0.06722x^7 \\ \rho_{\text{check}}(x) &= 0.10309x^{29} + 0.89691x^{30}. \end{aligned} \tag{236}$$

3640 message bits are encoded to get a coded block length of 4095. The channel is precoded PR-IV, and we assume AWGN.

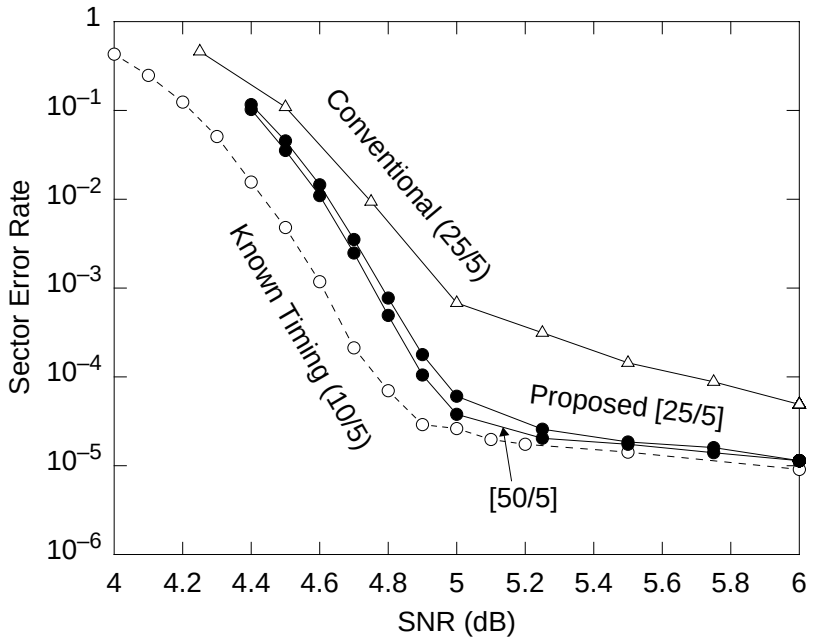

Figure 57: Rate-8/9 irregular LDPC code, moderate random walk, PR channel.

Figure 57 shows the sector error rate (SER) versus SNR for a moderate random walk case $\sigma_w/T = 0.5\%$ with $\alpha = 0.04$. A maximum of 10^6 packets were simulated for each SNR. 21 interpolation coefficients were used in (233). We use the notation

x/y to denote the scheduling of iterations, where we have y LDPC iterations before returning to the equalizer block, and we have a total of x such outer iterations, each involving timing recovery followed by turbo equalization. After the waterfall region, the proposed receiver almost achieves the performance of the system with perfect timing. The floor in the code performance is due to the presence of cycles in the graph of the code. The proposed timing recovery scheme performs well even in the presence of cycles.

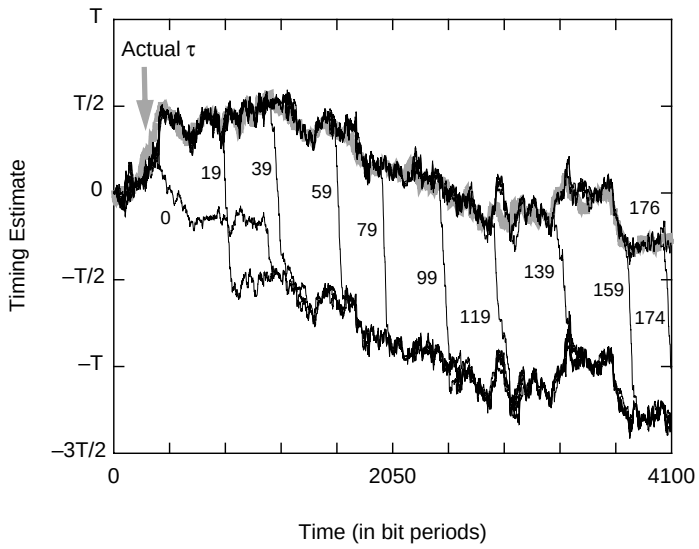

Figure 58: Automatic correction of cycle slips needs many iterations.

Next, we increase the random walk parameter to $\sigma_w/T = 1\%$ to study the effect of cycle slips on the performance of the system. As mentioned before, the proposed decoder corrects cycle slips on its own. This is illustrated for a sample packet at SNR = 5 dB in Figure 58. The gray curve represents the actual τ sequence, the curve labeled 0 shows $\hat{\tau}$ after the first pass of the PLL, and the other curves show $\hat{\tau}$ after the number of iterations indicated by the corresponding label. Automatic cycle slip correction is quite slow and, for this particular case, needs more than 175 iterations.

Convergence can be speeded up detecting the cycle slip early and correcting for it as discussed in Section 7.4.2. This method works by declaring a cycle slip whenever the

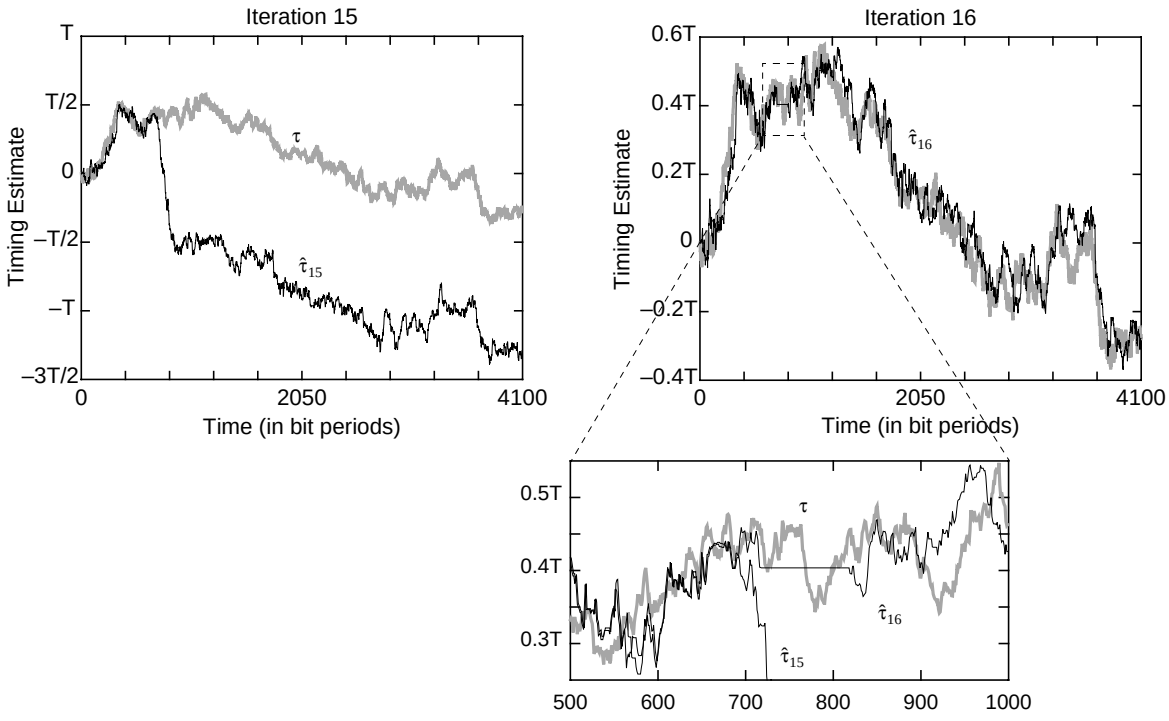

Figure 59: Cycle slip detection reduces the number of iterations needed.

slope of the timing estimate waveform exceeds a threshold. This process is illustrated for the same packet in Figure 59, with $H = 0.75T$ and $d = 100$ for the slip detection mechanism. With slip detection and correction, we correct the slip in this case by the 16^{th} iteration itself.

Figure 60 shows the performance of the same code for a severe random walk $\sigma_w/T = 1.0\%$, in which case we have cycle slips. In the presence of cycle slips, we employ the cycle slip detection/correction mechanism described below: Declare a slip whenever the magnitude of $\delta_k = \hat{\tau}_k - \hat{\tau}_{k-d}$ exceeds a given threshold H, for some delay d. If a cycle slip is detected, this is corrected by holding the timing waveform constant, for d symbols following the slip detection location, at the value just before the detection of the slip. A random walk is generated by summing zero-mean elements, and therefore, when a slip is detected, we correct it by holding the timing estimate constant at the value just before the slip occurred. After correcting for the slip, we run

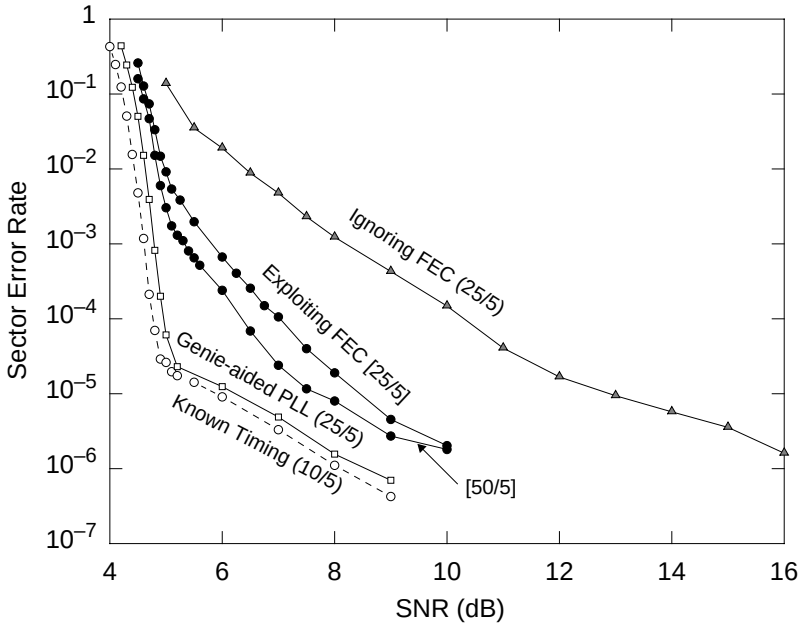

Figure 60: Rate-8/9 irregular LDPC code, severe random walk, PR channel.

the soft Mueller and Müller TED using memoryless soft decisions instead of decisions from the turbo equalizer. Other pertinent parameters are $\alpha = 0.055$, maximum of 10^7 blocks simulated, $d = 100$, $H = 0.75T$ and 21 interpolation coefficients. We observe a gain of around 4 dB at SER $= 10^{-5}$ when compared to the conventional system, and the enhanced turbo equalizer is less than 1 dB away from a conventional receiver with perfect timing.

7.4.5 High Rate Regular LDPC Code + Frequency Offset

Next, we consider a rate-8/9 $(3, 27)$-regular code of coded block length 4095, the parity-check matrix of which has 3 ones in each column and 27 ones in each row. The channel is precoded PR4 and we assume AWGN. In this case, we have only a frequency offset. We assume perfect acquisition, *i.e.*, $\tau_0 = 0$. The frequency offset parameter is $\Delta T = 0.2\%$. This is the same system considered for Figure 50. Within the duration of a block, the τ waveform varies by as much as $8T$, and therefore, cycle slips are quite likely. As opposed to the random walk case, where the difference between consecutive

elements of the τ sequence was zero-mean, the difference of consecutive elements of the τ sequence with a frequency offset is not zero-mean, and therefore the cycle slip correction mechanism has to be modified.

As before, we declare a slip whenever the magnitude of $\delta_k = \hat{\tau}_k - \hat{\tau}_{k-d}$ exceeds a given threshold H, for some delay d. If a slip is detected, then we use portions not affected by the slip to estimate the frequency offset by the least squares method, and use this estimate to correct the slipped region. After this is done, we revert to memoryless soft-slicer decisions for further timing recovery as opposed to the using soft decisions from the turbo equalizer.

If the number of iterations exceeds a certain threshold N_i, then we assume the presence of an undetected cycle slip, and use the following procedure:

- Construct a sequence $\{\delta_k\}$ where $\delta_k = \hat{\tau}_k - \hat{\tau}_{k-d}$,
- Compute the mean m_δ and the standard deviation σ_δ of the sequence $\{\delta_k\}$,
- Compute mean m_δ^p of those δ_k that lie in the interval $[m_\delta - \sigma_\delta, m_\delta + \sigma_\delta]$,
- Set $\hat{\Delta T} = m_\delta^p/d$,
- Use $\hat{\Delta T}$ to resample the whole block.

By doing this, in effect, we exclude the outliers (corresponding to the cycle slip) from the frequency offset computation. This procedure is more computationally intensive than the earlier one.

Figure 61 shows word (sector) error rate (SER) vs. SNR for the proposed system with $\alpha = 0.04$, $\beta = \alpha^2/4$, $H = 0.75T$, $d = 100$, with a maximum of 3×10^6 packets being simulated. The threshold N_i was 100 iterations, and after N_i iterations, the more complicated cycle-slip correction algorithm was implemented for at most 25 more iterations. At SER $= 1 \times 10^{-5}$, the proposed system gains around 3 dB when compared to the conventional system and is around 0.8 dB away from the system with perfect timing. Also, for this setting, the genie-aided system approaches the system with known timing for high SNR.

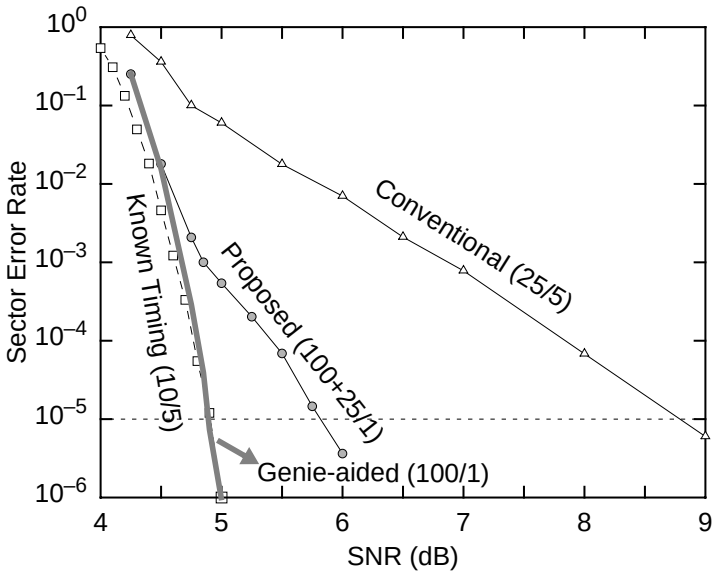

Figure 61: Rate-8/9 regular LDPC code, frequency offset, PR channel.

7.4.6 For BER/SER performance, PLL is adequate

In all the results so far, we see that iterative timing recovery using the PLL gets us close to the known timing performance, but we still have a performance gap, and at high SNR, this can be attributed to unresolved cycle slips. In Chapter 5, we derived optimal timing recovery methods that achieve the CRB. A natural question at this stage would be: what is the effect of using those timing recovery methods instead in the PLL in iterative timing recovery?

The timing recovery algorithms discussed in Chapter 5 were chosen to minimize the timing error variance, whereas, the metrics used in this chapter were the bit-error rate (BER) and the sector-error rate (SER). A large gap in the timing error variance performance translates to only a small gap in the BER or the SER curves as the turbo equalizer is capable of handling small timing errors very well, and it is only when the timing errors become large that we get bit errors. For the constant offset case, even though there is a 5 dB gap between the performance of the constant-gain PLL and the ML scheme (Figure 32) in terms of the timing error variance, the BER curves

for the two are virtually indistinguishable for the same set of parameters used to get Figure 32. In other words, when we are interested on the BER/SER performance of the system, a PLL is adequate.

In the case of the frequency offset, the Levenberg-Marquardt (LM) method described depends on the PLL for the first pass, and if the PLL has a cycle slip, the LM algorithm is not able to overcome the effects of the wrong initialization. The surface of the cost function is such that the receiver gets pushed into one of the many local minima. In cases where the PLL did not suffer a cycle slip, we observe the same effect where the gap in the timing error variance almost vanishes in the BER/SER domain. Therefore, even in this case, for BER/SER performance, the PLL performs adequately.

Finally, for the random walk case, the MAP estimator is a linear filter that acts on the output of the PLL, leading to a significant reduction in the timing error variance. The linear filter is a smoothing filter that performs localized weighted averaging on the PLL output, and in the presence of cycle slips, localized averaging is ineffective. Hence, we have a similar effect.

Therefore, for all the timing models considered here, in an iterative setting, a PLL performs adequately. This suggests that, in an iterative setting, it is the process of information exchange rather than the strict optimality of the constituent blocks that counts. This ties in with the excellent performance of RA codes. An RA code is a serial concatenation of a repetition code and an accumulator, both rather weak codes. With iterative decoding, RA codes have been shown to achieve capacity on certain channels, and approach capacity on others [29].

7.4.7 Optimal Acquisition and Iterative Tracking

We consider again the system of Section 7.4.5 with a rate-8/9 $(3, 27)$-regular code of coded block length 4095, the parity-check matrix of which has 3 ones in each

column and 27 ones in each row. The channel is precoded PR4 and we assume AWGN. We assume a frequency offset model with the frequency offset parameter being $\Delta T = 0.2\%$. To aid acquisition, the LDPC codeword is prefixed by a length-60 1100 periodic pattern, and followed by a length-60 random bit pattern. This conforms to the optimal acquisition arrangement derived in Chapter 6. These additional bits represent a 2.9% overhead in terms of the number of bits written into the channel per message packet. The acquisition strategy used here is the same as discussed in Section 6.5. During the iterations of the iterative timing recovery block, the cycle slip detection/correction strategy is the same as the one used in the Section 7.4.5.

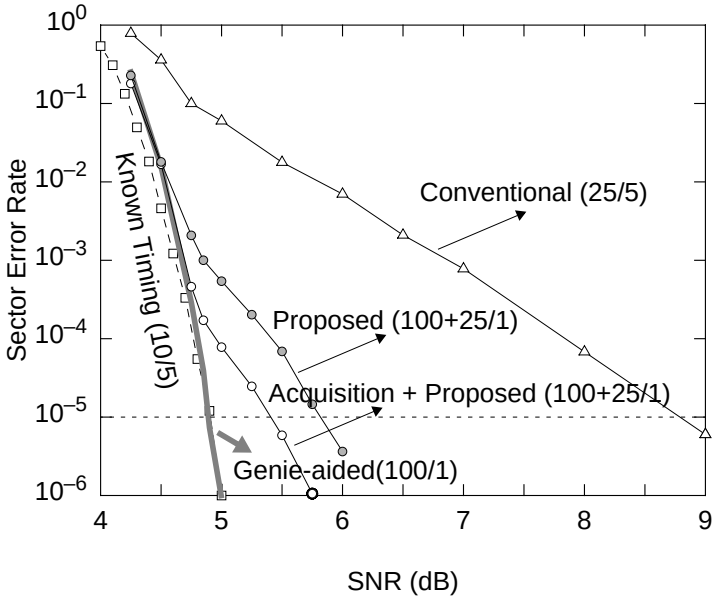

Figure 62: Performance with optimal acquisition and iterative tracking.

Figure 62 shows word (sector) error rate vs. SNR curves for this system with $\alpha = 0.04$, $\beta = \alpha^2/4$, $H = 0.75T$, $d = 100$, $N_i = 100$, with a maximum of 3×10^6 packets being simulated. The system with the optimal acquisition arrangement and iterative timing recovery during the tracking phase performs to within 0.4 dB of the genie-aided system at SER $= 1 \times 10^{-5}$, thus recovering most of the loss of the conventional system when compared to the system with known timing.

7.4.8 Complexity Comparison

In this section, we compare the computational complexity of the various schemes discussed in this chapter: a system where the timing offsets are known (System I); a genie-aided system where the timing recovery block has access to the transmitted symbols (System II); the conventional system that separates the turbo equalizer and the timing recovery (System III); the proposed joint timing recovery and turbo equalization system (System IV); and finally the system with optimal acquisition and iterative timing recovery during tracking (System V). We again consider the simulation setting of Sections 7.4.5 and 7.4.7. The measure of comparison is the average number of operations needed per coded bit. There are three kinds of operations involved in the receiver: addition (A), multiplication (M) and look-up (LU). (The logarithm and the sinc function are assumed to be implemented using a look-up table.)

First, we calculate the total number of such operations needed for each of the individual blocks: the timing recovery and interpolation block; the SISO equalizer based on the BCJR algorithm; and the LDPC decoder based on message passing.

- **Timing Recovery**: For each symbol index, the MM TED involves (2)M and (1)A. The PLL update involves (2)M and (2)A. Assuming we use K interpolation coefficients, we have (K)M, (K)A and (K)LU for the interpolation step. The first pass of the PLL does not involve interpolation, and therefore, its per-symbol complexity is (4)M and (3)A. For each symbol during a subsequent call of the timing recovery block, we have a total of $(K+4)$M, $(K+3)$A and (K)LU.

- **BCJR Equalizer**: Each γ computation involves (4)M, (1)A and (1)LU. Assuming N_i and N_o are the sizes of the input and the output alphabets, each stage involves N_iN_o gamma computations. Each α computation involves (N_i)A and (N_i)M. Similarly for each β. Each stage has Q αs and Q βs, where Q is the total number of states at any time instant in the trellis. Finally, the LLR

computation involves $(4Q+1)$M, $(2Q)$A and (1)LU. Putting it all together, we get $(4N_iN_o + 2QN_i + 4Q + 1)$M, $(N_iN_o + 2QN_i + 2Q)$A and $(N_iN_o + 1)$LU for each coded symbol in a call of the SISO equalizer.

- **LDPC decoder**: We consider a regular LDPC code. Let N_b be the total number of bit nodes and N_c be the total number of check nodes. Also, let d_b be the degree of the bit nodes and let d_c be the degree of the check nodes. The bit-to-check message computations involve $(N_b + 2d_bN_b)$A. The check-to-bit message computations involve $(2d_cN_c)$M, $(2d_cN_c)$A and $(2d_cN_c)$LU. Finally, the LLR evaluation involves (N_b)A. Adding it all up and dividing by N_b to get the complexity per bit, we get $(2d_b)$M, $(2(1 + 2d_b))$A and $(2d_b)$LU.

The three kinds of operations are combined into a single number as follows. We assume that A and LU are equivalent in terms of complexity and that an M can be implemented using 1A and 3LU using logarithms. For the LDPC-coded system under consideration with $K = 21$, $N_i = 2$, $N_o = 3$, $Q = 4$ and $d_b = 3$, we get the following complexity metrics for each of the constituent blocks:

- First pass timing recovery: 19,
- Subsequent passes of the timing recovery block: 145,
- BCJR equalizer: 265,
- LDPC decoder iteration: 44.

Next, by simulation, we compute the average number of times each of these blocks is called for the various systems. The stopping criterion for the iterations was as follows. At the end of each LDPC decoder iteration, we check whether all the parity-check equations are satisfied. If so, the receiver stops iterations. Else, iterations continue till a predetermined maximum number of iterations are reached. These

measured iteration numbers are then weighted by the complexity metrics listed above to get the final complexity measure, which is the total number of operations needed per coded bit for the different schemes.

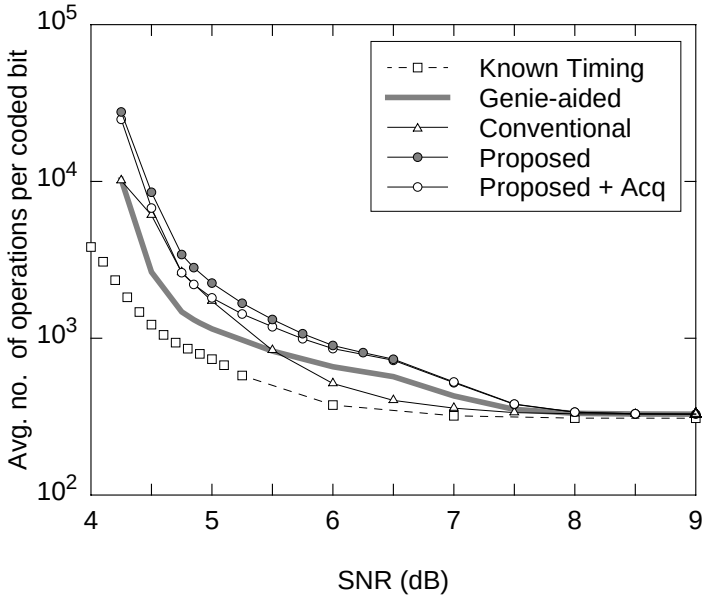

Figure 63: Comparing the complexity of the different schemes.

Figure 63 plots the complexity of the different systems as a function of SNR. The system parameters are the same as used to arrive at the performance plot of Figure 62. The system with known timing has the least complexity at all SNRs. For reasonably high SNRs, the complexity of the proposed iterative timing recovery scheme approaches that of the system with known timing. Asymptotically, as SNR increases, this difference is exactly the complexity of the first-pass PLL. The scheme with optimal acquisition and iterative timing recovery has lower complexity than the one with just iterative timing recovery. As SNR increases, Systems II, III, IV and V converge to the same complexity, which is the sum of the complexities of the first-pass PLL (19), the BCJR equalizer(265) and the LDPC decoder (44). Therefore, on an average, the proposed schemes are only marginally more complex than the conventional system and at the same time provides a distinct advantage in terms of

the SNR of operation as shown in Figure 62. The method used here to arrive at a single number denoting the complexity of the system is rather arbitrary, but even with different methods to quantify system complexity, we observe the same trends.

7.5 Summary

The conventional receiver that separates timing recovery and turbo equalization faces a large performance penalty even with moderately severe timing jitter models. Comparing its performance to that of the genie-aided system, we conclude that much of the loss is due to the poor quality of the decisions available to the timing recovery block. To solve this problem, we proposed iterative timing recovery where the better decisions after turbo equalization iterations are fed back to the timing recovery block. This led to improved quality of the samples, further improving the performance of the turbo equalizer. In all the cases considered, iterative timing recovery using the PLL-based timing recovery block recovered most of the performance loss of the conventional system when compared to the genie-aided system. In addition, the PLL-based system fares almost as well as the CRB-achieving timing recovery methods in an iterative setting.

An advantage of iterative timing recovery is the fact that it can correct cycle slips. However, the number of iterations needed to correct cycle slips can get large depending on the SNR and the timing models used. We present simple cycle slip detection/correction methods that significantly reduce the total number of iterations needed. The remaining SNR loss of iterative timing recovery is largely due to unresolved cycle slips. Both performance loss and complexity can be reduced further by modifying the acquisition algorithm. Splitting the preamble into two halves and placing these one at the beginning and one at the end reduces the occurrence of cycle slips, thus reducing the complexity of the system.

CHAPTER 8

CONCLUSION

In this thesis, we dealt with the problem of timing recovery at low SNR for channels beset with inter-symbol interference (ISI) and low signal-to-noise ratio (SNR). A good example of such a situation is the magnetic recording channel, especially in conjunction with the use of iteratively decodable codes that increase the recording density, thus increasing the level of ISI and reducing the SNR.

8.1 Main Contributions

The following are the main results in this thesis.

- We derived lower bounds on timing estimation error based on the Cramér-Rao bound for different timing offset models. These bounds were then used to evaluate the performance of the conventional PLL-based timing recovery method. [47]

- We developed block-processing timing recovery methods that outperformed the PLL-based one and, in some cases, achieved the Cramér-Rao bound. These are based on gradient-search, Newton's method and the maximum *a posteriori* estimation algorithms.

- We analyzed the performance of current acquisition algorithms, and derived the optimal placement of the preamble symbols to minimize the CRB on the timing estimation error. This optimal arrangement was also shown to reduce the occurrence of cycle slips.

- We proposed the iterative timing recovery algorithm for coded systems that use

iteratively decodable codes. Iterative timing recovery jointly performs timing recovery, equalization and decoding by embedding the timing recovery unit inside the turbo equalizer. This leads to significant reduction in the SNR of operation with marginal increase in complexity of the turbo equalizer iterations. [46] [45] [48] [4]

8.2 Future Work

- The problem of joint tracking and acquisition warrants further study. We assumed that we use correlation techniques to detect the start of transmission, thus limiting the initial timing offset to within a symbol duration. Lower bounds on performance and improved techniques for the general problem that includes detection of start of transmission, acquisition and tracking need to be developed.

- It is unclear whether the optimal preamble placement to minimize the CRB for the frequency case is still the optimal strategy in the presence of a random walk during the tracking phase. This needs to be analyzed further.

- Alternative methods that perform joint timing recovery, equalization and error-control decoding have been proposed and studied. Some of these are detailed in [31], [40], [4], [43], [50] and [17]. A framework unifying these various algorithms and the iterative timing recovery algorithm proposed here is desirable.

- The highly non-linear phenomenon of cycle slips has to be studied to improve the performance of iterative timing recovery. Analysis for simpler channel models is available in [64], but so far the combination of the channel and timing models considered here has been intractable. This analysis would hopefully help in dealing with cycle slips in a fashion more systematic than the current *ad hoc*, common sense ones employed in this thesis.

APPENDIX A

SOFT SLICER FOR THE PR-IV CHANNEL

In this appendix, we present the derivation of the soft slicer used in (15) in Section 2.2.3. We consider a memoryless soft slicer that uses observation r_k to output a soft decision $\tilde{d}_k = E[d_k|r_k]$. For the PR-IV channel, where d_k takes on values $-2, 0, +2$, $\tilde{d}_k$ becomes

$$\tilde{d}_k = 2\Pr[d_k = 2|r_k] - 2\Pr[d_k = -2|r_k], \tag{237}$$

where $\Pr[d_k = 2|r_k]$ is the probability that $d_k = 2$ given the observation r_k. Similarly for $\Pr[d_k = -2|r_k]$. Applying Bayes' rule, we get

$$\begin{aligned} \Pr[d_k = 2|r_k] &= \frac{p(r_k|d_k = 2)\Pr[d_k = 2]}{p(r_k)}, \\ \Pr[d_k = -2|r_k] &= \frac{p(r_k|d_k = -2)\Pr[d_k = -2]}{p(r_k)}, \end{aligned} \tag{238}$$

where $p(r_k|d_k = 2)$ is a shorthand notation for the *p.d.f.* of the observation conditioned on $d_k = 2$ evaluated at the observed value r_k. Similarly for $p(r_k|d_k = -2)$ and $p(r_k)$. $p(r_k)$ can be evaluated as follows:

$$p(r_k) = \frac{1}{4}p(r_k|d_k = -2) + \frac{1}{2}p(r_k|d_k = 0) + \frac{1}{4}p(r_k|d_k = 2). \tag{239}$$

With additive white Gaussian noise of variance σ^2, the conditional probabilities are

$$p(r_k|d_k = -2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(r_k + 2)^2}{2\sigma^2}\right\}, \tag{240}$$

$$p(r_k|d_k = 0) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(r_k)^2}{2\sigma^2}\right\} \text{ and} \tag{241}$$

$$p(r_k|d_k = 2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left\{-\frac{(r_k - 2)^2}{2\sigma^2}\right\}. \tag{242}$$

Combining these with (237), (238) and (239), we get

$$\tilde{d}_k = \frac{2\sinh(2r_k/\sigma^2)}{\cosh(2r_k/\sigma^2) + e^{2/\sigma^2}}. \tag{243}$$

APPENDIX B

INVERTING THE TOTAL INFORMATION MATRIX FOR A RANDOM WALK

In this appendix, we present the details regarding the inversion of the total information matrix of the random walk process given by (114),

$$\boldsymbol{J}_T = \frac{1}{\sigma_w^2}\begin{bmatrix} \lambda & -1 & 0 & \dots & 0 \\ -1 & \lambda & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & \lambda & -1 \\ 0 & \dots & 0 & -1 & \lambda - 1 \end{bmatrix},$$

where

$$\lambda = 2 + E_{h'}\frac{\sigma_w^2}{\sigma^2}. \tag{244}$$

To invert $\boldsymbol{J}_T$, we follow a three step procedure. First we do a Cholesky factorization of $\sigma_w^2\boldsymbol{J}_T$ to get $\sigma_w^2\boldsymbol{J}_T = \boldsymbol{L}\boldsymbol{L}^T$, where $\boldsymbol{L}$ is a lower triangular matrix. Then, we solve $\boldsymbol{L}\boldsymbol{C} = \boldsymbol{I}$ to get $\boldsymbol{C}$. Next, solve $\boldsymbol{L}^T\boldsymbol{B} = \boldsymbol{C}$ for $\boldsymbol{B}$, and finally, $[\boldsymbol{J}_T]^{-1} = \sigma_w^2\boldsymbol{B}$.

The matrix equation to be solved is

$$\begin{bmatrix} l_{00} & 0 & 0 & \dots & 0 \\ l_{10} & l_{11} & 0 & \dots & 0 \\ l_{20} & l_{21} & l_{22} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ l_{n0} & l_{n1} & l_{n2} & \dots & l_{nn} \end{bmatrix}\begin{bmatrix} l_{00} & l_{10} & l_{20} & \dots & l_{n0} \\ 0 & l_{11} & l_{21} & \dots & l_{n1} \\ 0 & 0 & l_{22} & \dots & l_{n2} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & l_{nn} \end{bmatrix} = \begin{bmatrix} \lambda & -1 & 0 & \dots & 0 \\ -1 & \lambda & -1 & \ddots & \vdots \\ 0 & \ddots & \ddots & \ddots & 0 \\ \vdots & \ddots & -1 & \lambda & -1 \\ 0 & \dots & 0 & -1 & \mu \end{bmatrix}, \tag{245}$$

where $\mu = \lambda - 1$ and $n = N - 1$.

If we consider (i,k) such that $|i-k|>1$, and equate the $(i,k)^{th}$ term on both sides, we get

$$\sum_{j=0}^{\min(i,k)} l_{ij}l_{kj} \;=\; 0. \tag{246}$$

Plugging in $i=0$, we get $l_{k0}=0$ for $k\geq 2$. Using this in the case with $i=1$, we get $l_{k1}=0$ for $k\geq 3$. Proceeding this way, we get $l_{kj}=0$ for $k\geq j+2$ where $0\leq j\leq N-4$. This means $\boldsymbol{L}$ is a Toeplitz matrix with only two non-zero diagonals.

Next, if we look at (i,k) such that $i-k=1$, and equate the $(i,k)^{th}$ term on both sides, we get

$$l_{i\,i-1}l_{i-1\,i-1} \;=\; -1 \quad\Longrightarrow\quad l_{i\,i-1} = -\frac{1}{l_{i-1\,i-1}}. \tag{247}$$

To simplify notation, we redefine $l_i = l_{ii}$, in which case, $\boldsymbol{L}$ takes the form

$$\boldsymbol{L} \;=\; \begin{bmatrix} l_0 & & & & \\ -\frac{1}{l_0} & l_1 & & & \\ & -\frac{1}{l_1} & l_2 & & \\ & & \ddots & \ddots & \\ & & & -\frac{1}{l_{n-1}} & l_n \end{bmatrix}, \tag{248}$$

where the entries not shown are all zero.

Finally, equating diagonal entries on both sides of Equation 245

$$l_i^2 = \begin{cases} \lambda & : \quad i=0 \\ \lambda - \frac{1}{l_{i-1}^2} & : \quad 1\leq i\leq N-2 \\ (\lambda-1) - \frac{1}{l_{i-1}^2} & : \quad i=N-1, \end{cases} \tag{249}$$

which gives a recursive definition of the matrix $\boldsymbol{L}$.

Next, we solve $\boldsymbol{LC}=\boldsymbol{I}$. We need to solve N matrix equations of the form

$$\boldsymbol{L}\boldsymbol{c}_i \;=\; \boldsymbol{e}_i, \tag{250}$$

where $\boldsymbol{c}_i$ and $\boldsymbol{e}_i$ represent the i^{th} column of $\boldsymbol{C}$ and $\boldsymbol{I}$ respectively. Each of these matrix equations has N scalar equations.

Let $\boldsymbol{c}_i = [c_{0i}\ c_{1i}\ \ldots\ c_{N-1\,i}]^T$. The first scalar equation in the i^{th} matrix equation ($i > 0$) gives $c_{0i} = 0$. Using this in the second equation, we get $c_{1i} = 0$. Proceeding thus, we can show that $c_{ji} = 0$ for $j < i$. The i^{th} equation gives $-c_{i-1\,i}/l_{i-1} + c_{ii}l_i = 1 \implies c_{ii} = 1/l_i$. For $j > i$, we get $c_{ji} = c_{j-1\,i}/(l_j l_{j-1})$. This recursion can be easily solved, and the final form for the c_{ji}s is given by

$$c_{ji} = \begin{cases} 0 & : \quad j < i \\ \frac{1}{l_i} & : \quad j = i \\ \frac{1}{l_i^2 l_{i+1}^2 \ldots l_{j-1}^2 l_j} & : \quad j > i. \end{cases} \tag{251}$$

Having defined $\boldsymbol{C}$, we next solve $\boldsymbol{L}^T\boldsymbol{B} = \boldsymbol{C}$. Writing this out, we have N matrix equations of the form

$$\begin{bmatrix} l_0 & -\frac{1}{l_0} & & & \\ & l_1 & -\frac{1}{l_1} & & \\ & & \ddots & \ddots & \\ & & & l_{N-2} & -\frac{1}{l_{N-2}} \\ & & & & l_{N-1} \end{bmatrix} \begin{bmatrix} b_{0i} \\ b_{1i} \\ \vdots \\ b_{N-2\,i} \\ b_{N-1\,i} \end{bmatrix} = \begin{bmatrix} c_{0i} \\ c_{1i} \\ \vdots \\ c_{N-2\,i} \\ c_{N-1\,i} \end{bmatrix}, \tag{252}$$

where $\boldsymbol{b}_i = [b_{0i}\ b_{1i}\ \ldots\ b_{N-1\,i}]^T$ is the i^{th} column of $\boldsymbol{B}$.

Beginning from the last scalar equation, corresponding to $j = N-1$, we get $b_{N-1\,i} = c_{N-1\,i}/l_{N-1}$. In general, we get $b_{ji} = b_{j+1\,i}/l_j^2 + c_{ji}/l_j$. Substituting the values for c_{ji} from Equation 251, we have

$$b_{ji} = \begin{cases} \frac{1}{l_i^2 l_{i+1}^2 \ldots l_{N-2}^2 l_{N-1}^2} & : \quad j = N-1 \\ \frac{b_{j+1\,i}}{l_j^2} + \frac{1}{l_i^2 l_{i+1}^2 \ldots l_{j-1}^2 l_j^2} & : \quad j \geq i \\ \frac{b_{j+1\,i}}{l_j^2} & : \quad j < i. \end{cases} \tag{253}$$

Notice that the recursion specified in Equation 253 deals with l_j only in the squared form. Therefore, to simplify notation further, define $f_{j+1} = l_j^2$. (The indices are related this way to simplify the calculations down the road.) The recursion

of Equation 249 can now be rewritten as

$$f_j = \begin{cases} \lambda & : \quad j = 1 \\ \lambda - \frac{1}{f_{j-1}} & : \quad 2 \leq j \leq N-1 \\ (\lambda - 1) - \frac{1}{f_{j-1}} & : \quad j = N. \end{cases} \tag{254}$$

Break up f_j into numerator n_j and denominator d_j so that $f_j = n_j/d_j$. If we now restrict ourselves to $2 \leq j \leq N-1$ and use Equation 254, we get

$$\begin{aligned} d_j &= n_{j-1} \\ n_j &= \lambda n_{j-1} - d_{j-1} \\ &= \lambda n_{j-1} - n_{j-2}, \end{aligned} \tag{255}$$

with initial conditions $n_1 = \lambda$ and $n_0 = 1$, given by $f_1 = n_1/d_1 = n_1/n_0 = \lambda/1$. This recursion has a solution given by

$$n_j = \frac{\eta^{j+2} - \eta^{-j}}{\eta^2 - 1}, \tag{256}$$

where

$$\eta = \frac{\lambda + \sqrt{\lambda^2 - 4}}{2} \tag{257}$$

is a zero of the characteristic polynomial $D^2 - \lambda D + 1$ associated with the recursion of Equation 255. Combining Equations 254 and 255,

$$f_j = \begin{cases} \frac{n_j}{n_{j-1}} & : \quad 1 \leq j \leq N-1 \\ \frac{n_j - n_{j-1}}{n_{j-1}} & : \quad j = N, \end{cases} \tag{258}$$

where we recognize the fact that f_N is got simply by subtracting 1 from the value for f_N given by the recursion.

From Equations 253 and 258,

$$b_{N-1\,i} = \frac{1}{f_{i+1} f_{i+2} \ldots f_{N-1} f_N} = \frac{n_i}{n_N - n_{N-1}}. \tag{259}$$

Proceeding backwards, we get

$$\begin{aligned} b_{N-2\,i} &= \frac{b_{N-1\,i}}{f_{N-1}} + \frac{1}{f_{i+1}f_{i+2}\cdots f_{N-2}f_{N-1}} \\ &= \frac{b_{N-1\,i}n_{N-2}}{n_{N-1}} + \frac{n_i}{n_{N-1}} \\ &= (\lambda-1)\frac{n_i}{n_N - n_{N-1}}. \end{aligned} \tag{260}$$

and in general,

$$b_{ji} = a_j \frac{n_i}{n_N - n_{N-1}} \tag{261}$$

for $j \geq i$, where a_j satisfies the recursion

$$a_{j-1} = \lambda a_j - a_{j+1} \tag{262}$$

with initial conditions $a_{N-1} = 1$ and $a_{N-2} = \lambda - 1$. This has the solution

$$a_j = \frac{\eta^{N-j} + \eta^{-N+1+j}}{\eta + 1} \tag{263}$$

with η as defined in Equation 257.

Next, we recognize the fact that $\boldsymbol{B}$, which is the inverse of a symmetric matrix, is symmetric. So, $b_{ij} = b_{ji}$, which takes care of the case when $j < i$. Therefore,

$$[\boldsymbol{J}_T]^{-1}_{ij} = \sigma_w^2[\boldsymbol{B}]_{ij} = \sigma_w^2 \frac{a_{\max(i,j)} n_{\min(i,j)}}{n_N - n_{N-1}} \tag{264}$$

for $0 \leq i, j \leq N-1$.

To get the CRB on the estimation error for the individual timing estimates τ_k, we need the diagonal elements of $[\boldsymbol{J}_T]^{-1}$. Combining Equations 244, 257, 263 and 264, and simplifying, the lower bound can be expressed as

$$\frac{E[(\hat{\tau}_i(\boldsymbol{r}) - \tau_i)^2]}{T^2} \geq \frac{[\boldsymbol{J}_1^{11}]^{-1}_{ii}}{T^2} = h \cdot f(i), \tag{265}$$

where

$$\begin{aligned} h &= \frac{\sigma_w^2}{T^2}\frac{\eta}{\eta^2 - 1} \quad \text{is the steady state value,} \\ f(i) &= \tanh\left((N + \frac{1}{2})\ln\eta\right)\left[1 - \frac{\sinh\left((N - 2i + \frac{1}{2})\ln\eta\right)}{\sinh\left((N+\frac{1}{2})\ln\eta\right)}\right], \\ \eta &= \frac{\lambda + \sqrt{\lambda^2 - 4}}{2} \quad \text{and} \quad \lambda = 2 + E_{h'}\frac{\sigma_w^2}{\sigma^2}. \end{aligned} \tag{266}$$

REFERENCES

[1] Adireddy, S., Tong, L., and Viswanathan, H., "Optimal placement of known symbols for frequency-selective flat-fading channels," *IEEE Transactions on Information Theory*, vol. 48, pp. 2338–2353, Aug. 2002.

[2] Adler, R. L., Coppersmith, D., and Hassner, M., "Algorithms for sliding block codes: An application of symbolic dynamics to information theory," *IEEE Transactions on Information Theory*, vol. IT-29, pp. 5–22, Jan. 1983.

[3] Bahl, L., Cocke, J., Jelinek, F., and Raviv, J., "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Theory*, vol. IT-20, pp. 284–287, Mar. 1974.

[4] Barry, J., Kavcic, A., McLaughlin, S., Nayak, A., and Zeng, W., "Iterative timing recovery," *IEEE Signal Processing Magazine*, vol. 21, pp. 89–102, Jan. 2004.

[5] Barry, J., Lee, E., and Messerschmitt, D., *Digital Communication*. Boston, Massachusetts: Kluwer Academic Publishers, third ed., 2004.

[6] Baum, L. E., Petrie, T., Soules, G., and Weiss, N., "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164–171, 1970.

[7] Benedetto, S., Divsalar, D., Montorsi, G., and Pollara, F., "Serial concatenation of interleaved codes: Performance analysis, design and iterative decoding," *IEEE Transactions on Information Theory*, vol. 44, pp. 909–926, May 1998.

[8] Bergmans, J. W. M., *Digital Baseband Transmission and Recording*, section 2.3, pp. 55–74. Boston, Massachusetts: Kluwer Academic Publishers, 1996.

[9] Berrou, C., Glavieux, A., and Thitimajshima, P., "Near Shannon limit error-correcting coding and decoding: Turbo codes," *Proceedings of the IEEE International Conference on Communications 1993*, vol. 2, pp. 1064–1070, May 1993.

[10] Budianu, C. and Tong, L., "Training symbol placement for packet transmissions under asynchronous influence," *Proceedings of the IEEE Workshop on Signal Processing Advances in Wireless Communications*, June 2003.

[11] Chiavaccini, E. and Vitetta, G. M., "A per-survivor phase-estimation algorithm for detection of PSK signals," *IEEE Transactions on Communications*, vol. 49, pp. 2059–2061, Dec. 2001.

[12] Christiansen, G. S., "Modeling of a PRML timing loop as a Kalman filter," *Proceedings of the IEEE Global Telecommunications Conference 1994*, vol. 2, pp. 1157–1161, Nov. 1994.

[13] Chung, S.-Y., Forney, G., Urbanke, R., and Richardson, T., "On the design of low-density parity-check codes within 0.0045 dB of the Shannon limit," *IEEE Communications Letters*, vol. 5, pp. 58–60, Feb. 2001.

[14] Chung, S.-Y., Urbanke, R., and Richardson, T., "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Transactions on Information Theory*, vol. 47, pp. 657–670, Feb. 2001.

[15] Cideciyan, R., Dolivo, F., Hermann, R., Hirt, W., and Schott, W., "A PRML system for digital magnetic recording," *IEEE Journal on Selected Areas in Communications*, vol. 10, pp. 38–56, Jan. 1992.

[16] Datta, S. and McLaughlin, S. W., "An enumerative method for run-length limited codes: Permutation codes," *IEEE Transactions on Information Theory*, vol. 45, pp. 2199–2204, Sept. 1999.

[17] Dauwels, J. and Loeliger, H.-A., "Joint decoding and phase estimation: an exercise in factor graphs," *Proceedings of the IEEE International Symposium on Information Theory, 2003*, pp. 231–231, July 2003.

[18] Del Re, E., Ronga, L. S., and Bartolozzi, M., "Robust iterative synchronization algorithm using Calabro Wolf perfect arrays," *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 2001*, vol. 4, pp. 2333–2336, May 2001.

[19] Dempster, A. P., Laird, N. M., and Rubin, D. B., "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society, Series B*, vol. 39, no. 1, pp. 1–38, 1977.

[20] Dong, M., Tong, L., and Sadler, B. M., "Optimal insertion of pilot symbols for transmissions over time-varying flat fading channels," *IEEE Transactions on Signal Processing*, vol. 52, pp. 1403–1418, May 2004.

[21] Driessen, P. F., "DPLL bit synchronizer with rapid acquisition using adaptive Kalman filtering techniques," *IEEE Transactions on Communications*, vol. 42, pp. 2673–2675, Sept. 1994.

[22] Ferrari, G., Anastasopoulos, A., Colavolpe, G., and Raheli, R., "Adaptive iterative detection for the phase-uncertain channel: limited-tree-search versus truncated-memory detection," *IEEE Transactions on Vehicular Technology*, vol. 53, pp. 433–442, Mar. 2004.

[23] Forney, G., "The Viterbi algorithm," *Proceedings of the IEEE*, pp. 268–278, Mar. 1973.

[24] Forney Jr., G. D. and Eyuboglu, M. V., "Combined equalization and coding using precoding," *IEEE Communications Magazine*, vol. 29, pp. 25–34, Dec. 1991.

[25] Gallager, R., "Low-density parity-check codes," *IRE Transactions on Information Theory*, vol. IT-8, pp. 21–28, Jan. 1962.

[26] Georghiades, C. and Snyder, D., "The expectation-maximization algorithm for symbol unsynchronized sequence detection," *IEEE Transactions on Communications*, vol. 39, pp. 54–61, Jan. 1991.

[27] Hamkins, J. and Divsalar, D., "Coupled receiver-decoders for low rate turbo codes," *Proceedings of the IEEE International Symposium on Information Theory, 2003*, pp. 381–381, July 2003.

[28] Immink, K. A. S., *Coding Techniques for Digital Recorders*. Englewood Cliffs, New Jersey: Prentice-Hall, 1991.

[29] Jin, H., Khandekar, A., and McEliece, R., "Irregular repeat-accumulate codes," *Proceedings of the 2nd International Symposium on Turbo Codes and Related Topics*, pp. 1–8, Sept. 2000.

[30] Jin, X. and Kavčić, A., "Cycle-slip detection using soft-output information," *Proceedings of the IEEE International Conference on Communications 2001*, vol. 9, pp. 2706–2710, June 2001.

[31] Kovintawevat, P., Erden, M. F., Kurtas, E., and Barry, J., "A new timing recovery architecture for fast convergence," *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS2003)*, vol. 2, pp. 13–16, May 2003.

[32] Kozierok, C., "PCGuide: Hard disk drives." Available online at http:www.pcguide.com/ref/hdd/index.htm.

[33] Loeliger, H.-A., Dauwels, J., Koch, V. M., and Korl, S., "Signal processing with factor graphs: examples," *Proceedings of the First International Symposium on Control, Communications and Signal Processing, 2004*, pp. 571–574, Mar. 2004.

[34] Lottici, V. and Luise, M., "Carrier phase recovery for turbo-coded linear modulations," *Proceedings of the IEEE International Conference on Communications 2002*, vol. 3, pp. 1541–1545, May 2002.

[35] Lottici, V. and Luise, M., "Embedding carrier phase recovery into iterative decoding of turbo-coded linear modulations," *IEEE Transactions on Communications*, vol. 52, pp. 661–669, Apr. 2004.

[36] Ma, X., Giannakis, G. B., and Ohno, S., "Optimal training for block transmissions over doubly selective wireless fading channels," *IEEE Transactions on Signal Processing*, vol. 51, pp. 1351–1366, May 2003.

[37] Marcus, B., Siegel, P. H., and Wolf, J. K., "Finite-state modulation codes for data storage," *IEEE Journal on Selected Areas in Communications*, vol. SAC-10, pp. 5–37, Jan. 1992.

[38] Mee, C. D. and Daniels, E. D., *Magnetic Recording.* New York: McGraw-Hill, 1987.

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

[40] Mielczarek, B. and Svensson, A., "Improved MAP decoders for turbo codes with non-perfect timing and phase synchronization," *Proceedings of the IEEE Vehicular Technology Conference 1999*, vol. 3, pp. 1590–1594, Sept. 1999.

[41] Mielczarek, B. and Svensson, A., "Joint adaptive rate turbo decoding and synchronization on Rayleigh fading channels," *Proceedings of the IEEE Vehicular Technology Conference 2001*, vol. 2, pp. 1342–1346, May 2001.

[42] Mielczarek, B. and Svensson, A., "Phase offset estimation using enhanced turbo decoders," *Proceedings of the IEEE International Conference on Communications 2002*, vol. 3, pp. 1536–1540, May 2002.

[43] Motedayen-Aval, I. and Anastasopoulos, A., "Polynomial-complexity noncoherent symbol-by-symbol detection with application to adaptive iterative decoding of turbo-like codes," *IEEE Transactions on Communications*, vol. 51, pp. 197–207, Feb. 2003.

[44] Mueller, K. and Müller, M., "Timing recovery for digital synchronous data receivers," *IEEE Transactions on Communications*, vol. com-24, pp. 516–531, May 1976.

[45] Nayak, A., Barry, J., and McLaughlin, S., "Joint timing recovery and turbo equalization for coded partial response channels," *IEEE Transactions on Magnetics*, vol. 38, pp. 2295–2297, Sept. 2002.

[46] Nayak, A., Barry, J., and McLaughlin, S., "Joint timing recovery and turbo equalization for partial response channels," *Proceedings of the IEEE International Conference on Magnetics (Intermag) 2002*, p. BR03, Apr. 2002.

[47] Nayak, A., Barry, J., and McLaughlin, S., "Lower bounds for the performance of iterative timing recovery at low SNR," *Proceedings of the Fifteenth International Symposium on Mathematical Theory of Networks and Systems (MTNS)*, pp. WM2–5, Aug. 2002.

[48] Nayak, A., Barry, J., and McLaughlin, S., "Iterative timing recovery and turbo equalization," *Proceedings of the 3rd International Symposium on Turbo Codes and Related Topics*, p. M01, Sept. 2003.

[49] Negi, R. and Cioffi, J., "Pilot tone selection for channel estimation in a mobile OFDM system," *IEEE Transactions on Consumer Electronics*, vol. 44, pp. 1122–1128, Aug. 1998.

[50] Noels, N., Herzet, C., Dejonghe, A., Lottici, V., Steendam, H., Moeneclaey, M., Luise, M., and Vandendorpe, L., "Turbo synchronization : an EM algorithm interpretation," *Proceedings of the IEEE International Conference on Communications, 2003*, vol. 4, pp. 2933–2937, May 2003.

[51] Nuriyev, R. and Anastasopoulos, A., "Analysis of joint iterative decoding and phase estimation for the noncoherent AWGN channel, using density evolution," *Proceedings of the IEEE International Symposium on Information Theory, 2002*, p. 168, July 2003.

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

[53] Patapoutian, A., "On phase-locked loops and Kalman filters," *IEEE Transactions on Communications*, vol. 47, pp. 670–672, May 1999.

[54] Raphaeli, D. and Zarai, Y., "Combined turbo equalization and turbo decoding," *Proceedings of the IEEE Global Telecommunications Conference 1997*, vol. 2, pp. 639–643, Nov. 1997.

[55] Richardson, T., Shokrollahi, A., and Urbanke, R., "Design of capacity-approaching irregular low-density parity-check codes," *IEEE Transactions on Information Theory*, vol. 47, pp. 619–637, Feb. 2001.

[56] Roh, G., Lee, Y., and Kim, B., "Optimum phase-acquisition technique for charge-pump PLL," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, pp. 729–740, Sept. 1997.

[57] Roweis, S., "Levenberg-Marquardt optimization." Available online at http://www.cs.toronto.edu/∼roweis/notes/lm.pdf.

[58] Scharf, L. L., *Statistical Signal Processing: Detection, Estimation, and Time Series Analysis*, p. 230. New York: Addison-Wesley, 1990.

[59] Scharf, L. L. and McWhorter, L., "Geometry of the Cramér-Rao bound," *Signal Processing*, vol. 31, pp. 1–11, Apr. 1993.

[60] Simon, M. K. and Villnrotter, V. A., "Iterative information-reduced carrier synchronization using decision feedback for low-SNR applications," *The NASA Telecommunications and Data Acquisition Progress Report 42-130*, June 1997.

[61] Souvignier, T., Friedmann, A., Oberg, M., Siegel, P., Swanson, R., and Wolf, J., "Turbo decoding for PR4: Parallel vs. serial concatenation," *Proceedings of the IEEE International Conference on Communications 1999*, vol. 3, pp. 1638–1642, June 1999.

[62] Steendam, H., Noels, N., and Moeneclaey, M., "Iterative carrier phase synchronization for low-density parity-check coded systems," *Proceedings of the IEEE International Conference on Communications, 2003*, vol. 5, pp. 3120–3124, May 2003.

[63] Van Trees, H. L., *Detection, Estimation, and Modulation Theory*, vol. 1, chapter 2, pp. 72–85. New York: John Wiley and Sons, Inc., first ed., 1968.

[64] Van Trees, H. L., *Detection, Estimation, and Modulation Theory*, vol. 2, chapter 3, pp. 37–84. New York: John Wiley and Sons, Inc., first ed., 1971.

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

[66] Wiberg, N., *Codes and Decoding on General Graphs*. PhD dissertation, University of Linköping, Department of Electrical Engineering, 1996.

[67] Wicker, S. B., *Error Control Systems for Digital Communication and Storage*. Upper Saddle River, New Jersey: Prentice Hall, 1995.

[68] Xiao, P. and Ström, E., "Synchronization algorithms for iterative demodulated M-ary DS-CDMA systems," *Proceedings of the IEEE Global Telecommunications Conference, 2001*, vol. 2, pp. 1371–1375, Nov. 2001.

[69] Zeng, W. and Kavcic, A., "MAP detection in noisy channels with synchronization errors (including the insertion/deletion channel)," *Proceedings of the IEEE International Symposium on Information Theory, 2003*, pp. 232–232, July 2003.

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

[71] Zhang, L. and Burr, A. G., "APPA symbol timing recovery scheme for turbo codes," *Proceedings of the 13th IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, 2002*, vol. 1, pp. 44–48, Sept. 2002.

[72] Zhao, Q., Kim, H., and Stuber, G. L., "Adaptive iterative phase synchronization for serially concatenated continuous phase modulation," *Proceedings of the IEEE Military Communications Conference, 2003*, vol. 1, pp. 78–83, Oct. 2003.

VITA

Aravind Nayak was born in Davangere, India in 1978, and attended high school in Hyderabad, India. He joined the Indian Institute of Technology, Madras in 1995 for his undergraduate education, and obtained his Bachelor in Technology (B. Tech.) degree in Electrical Engineering in July 1999. He also received a Master of Science (M.S.) from the School of Electrical and Computer Engineering at the Georgia Institute of Technology, Atlanta in December 2000. He has been a student member of the IEEE since 1999. He completed the requirements for the degree of Doctor of Philosophy (Ph. D.) at the Georgia Institute of Technology in June 2004.