Iterative Timing Recovery

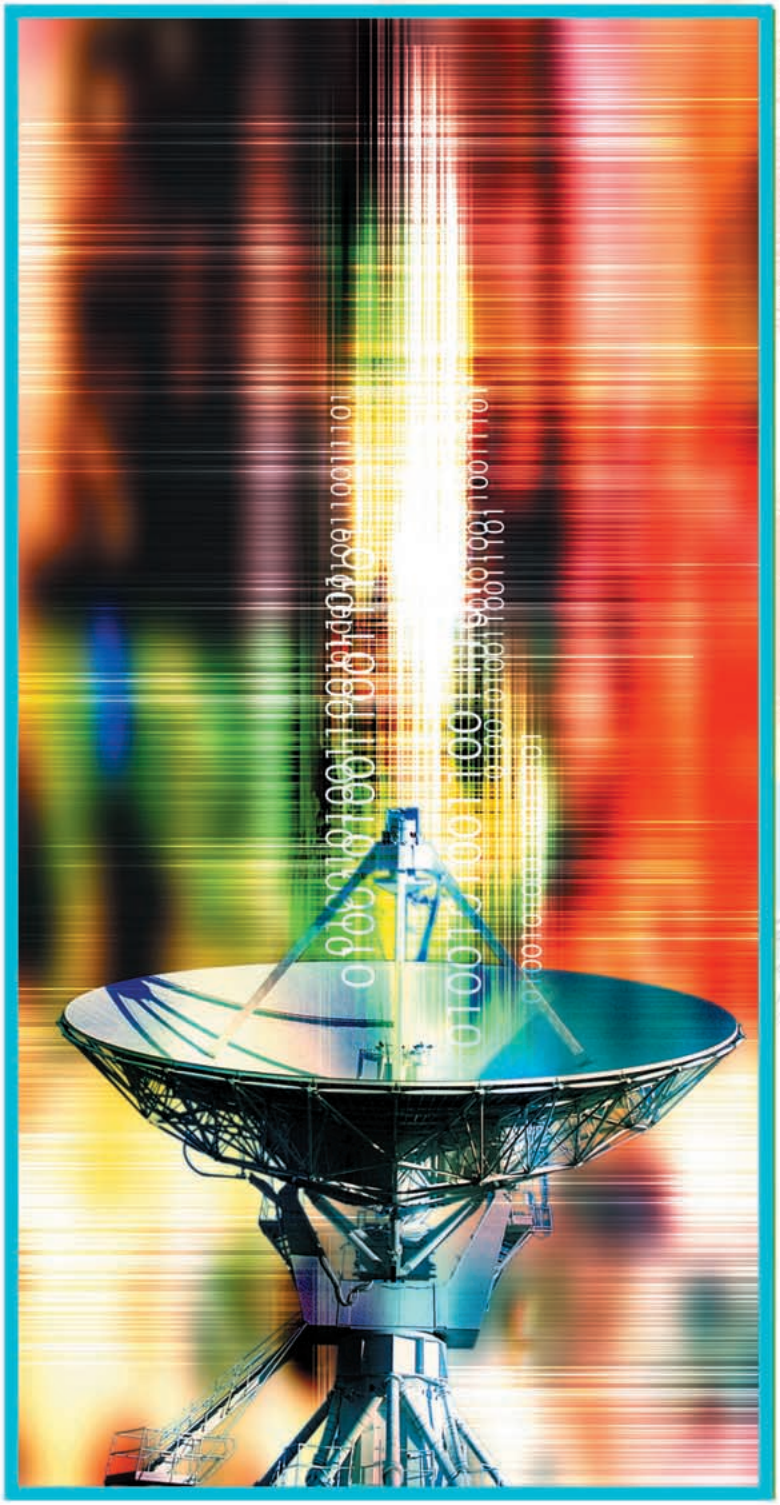
Methods for implementing timing recovery in cooperation with iterative error-control decoding.

John R. Barry,
Aleksandar Kavčić,
Steven W. McLaughlin,
Aravind Nayak, and
Wei Zeng

The last decade has seen the development of iteratively decodable error-control codes of unprecedented power, whose large coding gains enable reliable communication at very low signal-to-noise ratio (SNR). A by-product of this trend is that timing recovery must be performed at an SNR lower than ever before. Conventional timing recovery ignores the presence of error-control coding and is thus doomed to fail when the SNR is low enough. This article describes iterative timing recovery, a method for implementing timing recovery in cooperation with iterative error-control decoding so as to approximate a more complicated receiver that jointly solves the timing recovery and decoding problems.

Timing Recovery Problem

At some point in a digital communications receiver, an analog waveform must be sampled. Sampling at the right times is critical to achieving good overall performance. The process of synchronizing the

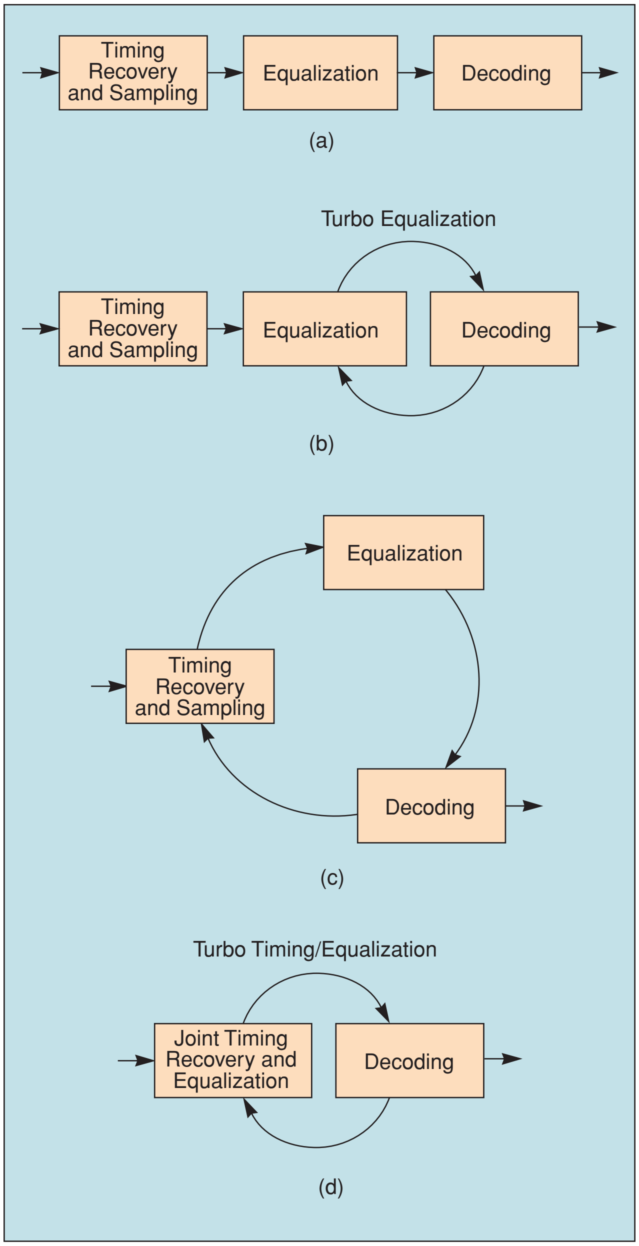


© DIGITAL VISION, LTD.

1053-5888/04/$20.00©2004IEEE

sampler with the pulses of the received analog waveform is known as timing recovery.

The timing recovery problem might not be difficult in isolation, but a practical receiver must not only perform timing recovery but also detection of the transmitted message, which involves such tasks as equalization and error-control decoding. In doing so it must contend with not only uncertainty in the timing of the pulses but also with additive noise and intersymbol interference (ISI). In principle, one could formulate the problem of jointly determining the maximum-likelihood estimates of the timing offsets and the message bits. The solution to this problem would naturally perform the tasks of timing recovery, equalization, and decoding jointly. Unfortunately, the complexity would be prohibitive. Instead, a conventional receiver performs these tasks separately and sequentially, as illustrated in Figure 1(a). Specifically, a conventional timing recovery scheme ignores error-control coding, assuming instead that the transmitted symbols are mutually independent.


Timing Recovery and Sampling
Equalization
Decoding
(a)
Turbo Equalization
Timing Recovery and Sampling
Equalization
Decoding
(b)
Equalization
Timing Recovery and Sampling
Decoding
(c)
Turbo Timing/Equalization
Joint Timing Recovery and Equalization
Decoding
(d)

▲ *1. Conventional timing recovery with (a) conventional equalization and with (b) turbo equalization. Iterative timing recovery with (c) separate timing recovery and equalization and (d) joint timing recovery and equalization.*

The conventional separation approach of Figure 1(a) works reasonably well at a high SNR but not at the low SNRs supported by capacity-approaching iteratively decodable codes. As an extreme example [1], a recent rate-1/31 code designed for a deep-space application will operate reliably when the SNR is less than −15 dB! At this low SNR, a timing recovery scheme that ignores the code will likely fail. In this article, we describe an iterative method for jointly performing the tasks of timing recovery and error-control decoding with complexity comparable to a conventional receiver.

Perhaps the first iterative timing recovery method is due to Georghiades and Snyder, who applied the expectation-maximization (EM) algorithm for the case of constant offset with an ideal, uncoded system [2]. Since then there have been many extensions to account for ISI, time-varying offsets, and error-control coding [1], [3]–[5]. There are also close ties between timing recovery and carrier recovery, and so the recent work on using iterative concepts to estimate carrier phase is relevant [1], [6]–[8].

In this article, we present some emerging iterative timing recovery schemes. In Figure 1(a), a classical (noniterative) receiver is shown, where the signal flow is unidirectional from the timing recovery unit to the decoding unit. The advent of turbo codes [9] has prompted the turbo equalization architecture of Figure 1(b), where the signal flow from the timing recovery unit is unidirectional while there is an iterative exchange between the equalizer and the decoder. The scenario of Figure 1(b) will not be further examined in this article since it is covered in detail in a different article of this issue [10]. Instead, we focus on two methods that extend turbo equalization to include timing recovery iterations. The first such method is schematically depicted in Figure 1(c), where a timing recovery step has been added to each iteration of a turbo equalizer. The second method [depicted in Figure 1(d)] merges the timing recovery unit and the soft-output equalizer into a single unit that performs joint timing recovery and soft-output equalization in a manner similar to the BCJR [11] or Baum-Welch algorithm [12].

Classical Timing Recovery

In this section we present a brief overview of the most widely used method for timing recovery; namely, a decision-directed, phase-locked loop.

A digital transmitter conveys information by modulating the amplitudes of a sequence of pulses. In theory, these pulses are transmitted at a fixed rate—the baud rate $1/T$—that is known to the receiver. In practice, however, the baud rate may vary slowly with time, or the channel may introduce jitter because of Doppler effects or other nonidealities. Moreover, manufacturing imperfections may cause the transmitter and receiver clock frequencies to differ by a fraction of a percent. The reality is that the receiver never knows the precise arrival times of the pulses.

We consider the following simple model for the received waveform $r(t)$

$$r(t) = \sum_k a_k h\left(t - kT - \tau_k\right) + n(t), \qquad (1)$$

where $a_k \in \{\pm 1\}$ is the kth binary symbol (or amplitude), $h(t)$ is the received pulse shape, T is the signaling interval anticipated by the receiver, and $n(t)$ is additive Gaussian noise. The uncertainty in the timing is captured by the offset parameter τ_k, defined as the difference between the actual and expected arrival time of the kth pulse. For example, when the transmitter and receiver clock frequencies differ, τ_k increases linearly with time.

Our restriction to binary alphabets is sufficient to capture the gist of the timing recovery problem. Only minor modifications are needed to handle higher-order complex alphabets.

The optimal sampling times are $\{kT + \tau_k\}$. The job of a timing recovery scheme is to estimate the timing offsets before sampling. Nearly all existing timing recovery schemes are based on a *phase-locked loop* (PLL), which is easily described in terms of the following key definitions:

Let $\hat{\tau}_k$ denote the receiver's estimate of τ_k.
Let $\epsilon_k = \tau_k - \hat{\tau}_k$ denote the residual error in this estimate.
Let $\hat{\epsilon}_k$ denote the receiver's estimate of ϵ_k.

A second-order PLL estimates τ_k according to the recursion

$$\hat{\tau}_{k+1} = \hat{\tau}_k + \alpha \hat{\epsilon}_k + \beta \sum_{i=0}^{k} \hat{\epsilon}_i, \qquad (2)$$

where α and β are the proportional and integral step sizes, respectively. In other words, the offset estimate is found by accumulating the output of a loop filter with transfer function $\alpha + \beta/(1 - z^{-1})$ and input $\hat{\epsilon}_k$. A block diagram of the PLL is shown in Figure 2.

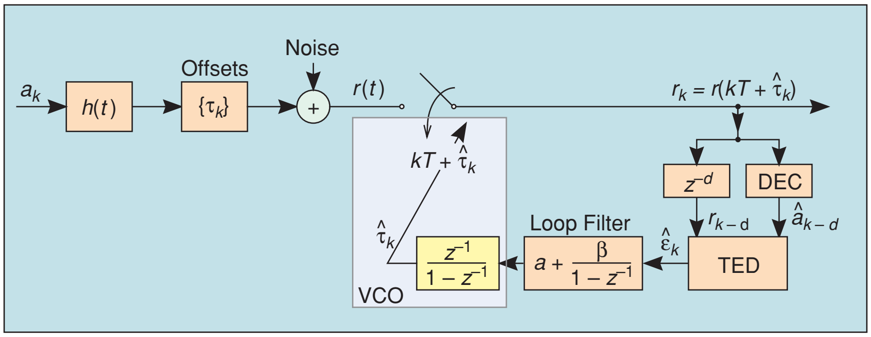
The PLL recursion (2) can be motivated heuristically as follows. Consider the case when β is zero, in which case (2) reduces to a first-order PLL. At time k, the receiver knows its current estimate $\hat{\tau}_k$, and it also has access to an estimate $\hat{\epsilon}_k$ of the residual error $\tau_k - \hat{\tau}_k$. If this estimate were somehow known to be precisely accurate, then the receiver could cancel the remaining residual timing error in only one step by choosing $\hat{\tau}_{k+1} = \hat{\tau}_k + \hat{\epsilon}_k$. In practice, however, $\hat{\epsilon}_k$ is a very noisy estimate of $\tau_k - \hat{\tau}_k$, and by attenuating $\hat{\epsilon}_k$ by a factor $\alpha < 1$, the PLL essentially low-pass filters the noise. The downside of this attenuation is that it will take more than one step to correct for the residual error, which implies that the PLL will be less agile when tracking a time-varying offset. In practice, the step size $\alpha \in (0, 1)$ is chosen to trade off the opposing goals of robustness to noise and agility. A nonzero value of β takes into account long-term trends in τ_k, giving a second-order PLL the ability to track a frequency offset with zero average steady-state error.

The device that generates $\hat{\epsilon}_k$ is called a timing-error detector (TED), and its effectiveness is key to the overall performance. A popular choice is the Mueller and Müller (M&M) TED [13]

$$\hat{\epsilon}_k = r_k a_{k-1} - r_{k-1} a_k, \qquad (3)$$

where $r_k = r(kT + \hat{\tau}_k)$. This TED is most accurate when there is no ISI, so that $h(kT) = 0$ for nonzero integers k. Together, (2) and (3) define the conventional approach to timing recovery.

The M&M TED of (3) requires knowledge of the transmitted symbols $\{a_k\}$, which is generally available to the receiver only during an initial start-up or training phase. In practice, a receiver without training will implement timing recovery in decision-directed mode by replacing $\{a_k\}$ in (3) by tentative decisions. The decision device that generates these decisions will invariably introduce a processing delay, which we denote by d. Therefore, as shown in Figure 2, a decision-directed PLL is driven by $\{\hat{a}_{k-d}\}$ instead of $\{a_k\}$. The performance of a decision-directed PLL depends critically on two parameters: 1) reliability of the decisions and 2) processing delay d of the decision device. There is a fundamental tradeoff between these two parameters, since reliability can generally be increased at the expense of processing delay. The importance of



▲ *2. The traditional approach to timing recovery is based on a decision-directed PLL. DEC stands for decision device, whose processing delay is d. In practice, the shaded block is implemented by passing the loop filter output through a voltage controlled oscillator (VCO). Here we model the VCO as a simple integrator.*

Conventional approaches to timing recovery fail at low SNR because they ignore error-control coding.

reliability is obvious, since incorrect decisions will drive the timing estimates away from their true values. The importance of delay stems from the well-known fact that a delay in a feedback loop can lead to instability. Specifically, a linearized analysis of (2) with $\hat{\epsilon}_k$ replaced by ϵ_{k-d} shows that an increase in the delay d moves the closed-loop poles closer to the unit circle, so that the step size parameter α must be decreased to maintain stability. The decreased step size makes the system less agile to time-varying timing offsets. Intuitively, long-delayed decisions are useless in the face of time-varying offsets, regardless of how reliable they might be, so this loss of agility is not surprising.

At one extreme, an instantaneous (zero-delay) decision can be extracted by quantizing r_k directly, leading to the following when there is no ISI:

$$\hat{a}_k = \text{sign}(r_k). \tag{4}$$

Such decisions have zero processing delay but can be very unreliable. At low SNR, better performance can be achieved by replacing this hard decision by a so-called soft decision $\tilde{a}_k = \text{E}[a_k | r_k]$, which for the binary alphabet and additive Gaussian noise with variance σ^2 reduces to

$$\begin{aligned}\tilde{a}_k &= \text{E}[a_k | r_k] \\ &= \frac{\frac{1}{2} \cdot 1 \cdot \exp \frac{-(r_k-1)^2}{2\sigma^2} + \frac{1}{2} \cdot (-1) \cdot \exp \frac{-(r_k+1)^2}{2\sigma^2}}{\frac{1}{2} \cdot \exp \frac{-(r_k-1)^2}{2\sigma^2} + \frac{1}{2} \cdot \exp \frac{-(r_k+1)^2}{2\sigma^2}} \\ &= \tanh \frac{r_k}{\sigma^2}.\end{aligned} \tag{5}$$

At the other extreme, an entire frame of samples $\{r_k\}$ can be passed on to an equalizer or channel decoder that can increase reliability by exploiting the structure of any channel ISI or forward-error control coding. The resulting decisions might be very reliable, but the excessive decoding delay precludes their use in real-time analog timing recovery. Fortunately, there is also a middle ground. For example, tentative decisions with delay d can be recovered from a Viterbi-based equalizer that backtracks the survivor memory for d stages. These decisions will be suboptimal when d is small.

We can avoid the problem of detection delay altogether if we adopt an off-line strategy known as interpolative timing recovery [14]. Instead of using the voltage-controlled oscillator (VCO) of a PLL to drive the sampling times, as shown in Figure 2, we could instead sample $r(t)$ asynchronously and uniformly with a free-running clock at a rate fast enough to avoid aliasing. This might be the baud rate, a few percent higher than the baud rate, or twice the baud rate, depending on the amount of excess bandwidth and the severity of the worst-case frequency offset. The asynchronous samples can be used to estimate what the sampling times should have been, and then digital interpolation can be used to effectively resample at the proper times. It can be shown [15] that the interpolative strategy is mathematically equivalent to resampling, yet it is much simpler because it does not require any analog-to-digital converters—only logic circuits. This interpolative strategy opens up the door to sophisticated detectors with large decoding delay that would otherwise be impractical. The idea is explored in the next section.

Iterative Timing Recovery

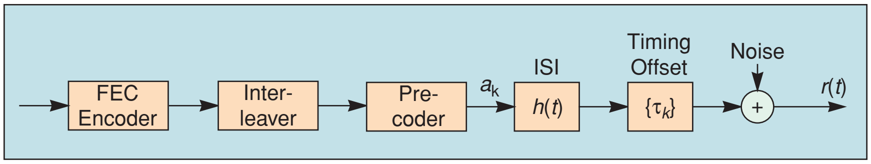
In this section we describe iterative timing recovery for the transmitter and channel shown in Figure 3, where the channel output is described by the model (1). The transmitted symbols $\{a_k\}$ are the result of coding, interleaving, and precoding the information bits.

In principle, the receiver would like to perform the tasks of timing recovery, equalization, and decoding jointly so as to maximize the joint likelihood function. Although the complexity of a direct implementation is prohibitive, an iterative receiver provides a low-complexity approximation to the joint solution. Turbo equalization is already a well-known technique for iteratively approximating the joint-ML equalizer and decoder [10]. In the following we describe how timing recovery can be added to the mix.

Algorithm Description

The gist of iterative timing recovery is simple; rather than performing timing recovery once, without any help from the decoder, timing recovery is performed multiple times while interacting with the decoder.

One way to implement iterative timing recovery is to follow the recursion depicted in Figure 4. First, the receiver estimates the timing in a conventional manner, ignoring the ISI and code. The timing estimates are then passed to the symbol estimator that uses them to sample, equalize, and decode. The output of the symbol estimator is a sequence of soft esti-



▲ *3. A transmitter consisting of a forward error-control encoder, an interleaver, and a precoder. The channel introduces intersymbol interference, a time-varying timing offset, and additive noise.*

mates of the transmitted symbols. Based on these soft decisions, timing estimation is repeated, generating new timing estimates that are more accurate than the originals. For example, this timing reestimation might be implemented using a PLL whose M&M TED (3) is driven by the soft decisions from the decoder. The process is then repeated: the new timing estimates are used to sample, equalize, and decode, and the resulting soft decisions are used to generate the next set of timing estimates. The hope is that each set of symbol estimates will improve the timing estimates, which will in turn improve the next set of symbol estimates, so that eventually the algorithm will converge to the optimal solution of the joint-maximum-likelihood problem. For certain channel models this recursion reduces to an instance of the EM or Baum–Welch algorithm [12], in which case the recursion provably converges to a local maximum of the joint likelihood function. It is important to note that the timing estimator and symbol estimator are standard blocks that would be needed anyway; the only new idea is to have them cooperate in a simple manner.

Although useful in concept, the iterative picture in Figure 4 is somewhat misleading because it overstates the complexity of iterative timing recovery. Suppose the symbol estimator is implemented using an iterative turbo equalizer that already dominates the receiver complexity. Each iteration of the turbo equalizer consists of one call to a soft-output equalizer followed by one call to a soft-output, error-control decoder [10]. Therefore, if the recursion of Figure 4 is iterated K times then it appears that the iterative timing-recovery scheme effectively increases complexity by a factor of K.

Fortunately, there is an efficient way to implement iterative timing recovery that is only marginally more complex than the conventional approach. The basic idea is to start with a conventional turbo equalizer and insert a timing estimation step in between each iteration of the turbo equalizer, thus transforming the two-way iterative procedure of Figure 4 into the three-way iterative procedure of Figure 1(c). In pseudocode, iterative timing recovery can be summarized as follows:

```
sample the waveform using a decision-
directed PLL to get waveform samples
$\{r_k\}$
for $i = 1 : K$,
  equalize samples $\{r_k\}$ (using the BCJR
  [11] algorithm) and pass soft
  outputs to decoder
  decode to get estimates of the
  transmitted symbols $\tilde{a}_k$
  estimate timing error using PLL
  interpolate samples (equivalently
  resample waveform) to get new
  samples $\{r_k\}$
end
```

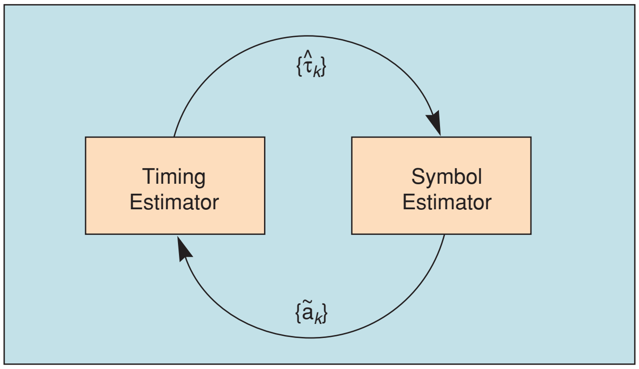
A benefit of iterative timing recovery is that it automatically corrects for cycle slips.

In this way, soft decisions from each turbo equalizer iteration are used to improve the timing estimates and resample the received signal before proceeding with the next turbo equalizer iteration [4]. As long as the complexity of the timing estimator and interpolation process are negligible compared to the complexity of the soft-output equalizer and soft-output decoder (which are typically based on the BCJR algorithm), the overall scheme will have comparable complexity to a conventional receiver.

Performance Results

We now present some numerical results for the partial-response channel $h(t) = \sum_k h_k g(t - kT)$, where $g(t) = \sin(\pi t/T)/(\pi t/T)$ is an ideal zero-excess-bandwidth Nyquist pulse. Specifically, we assume $H(D) = \sum_k h_k D^k = 1 - D^2$, a channel known as PR4 in the context of magnetic recording. The timing recovery problem is relatively easy to solve in this case, because we can view the received signal as the output of an ideal ISI-free channel when the input is a sequence of multilevel symbols $b_k = \sum_n h_n a_{k-n}$. Thus, the only modification necessary to the conventional decision-directed timing recovery system of Figure 2 is to modify the decision device to estimate the multilevel symbol b_k instead of the original binary symbol a_k. For the PR4 channel, the multilevel symbol takes on one of three values, $b_k = a_k - a_{k-2} \in \{0, \pm 2\}$. An instantaneous soft estimate of b_k given r_k is $\tilde{b}_k = \mathrm{E}[b_k | r_k]$, which reduces to (with Gaussian noise with variance σ^2), [4]

$$\tilde{b}_k = \frac{2\sinh\frac{2r_k}{\sigma^2}}{\cosh\frac{2r_k}{\sigma^2} + \exp\frac{2}{\sigma^2}}, \qquad (6)$$



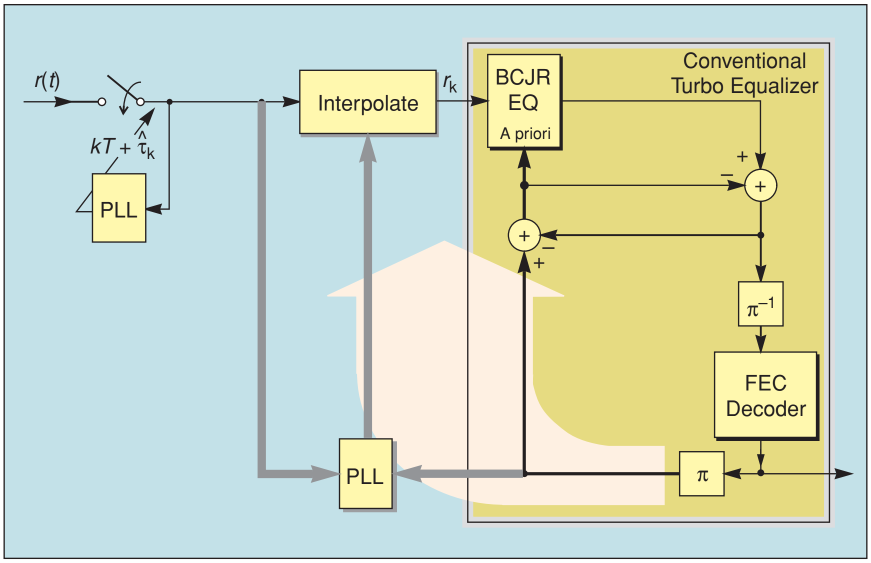
▲ *4. An EM-like recursion between a timing estimator (which ignores the ISI and code) and a symbol estimator (which performs equalization and decoding while ignoring the residual timing offset).*

where this formula can be obtained by arguments similar to (5).

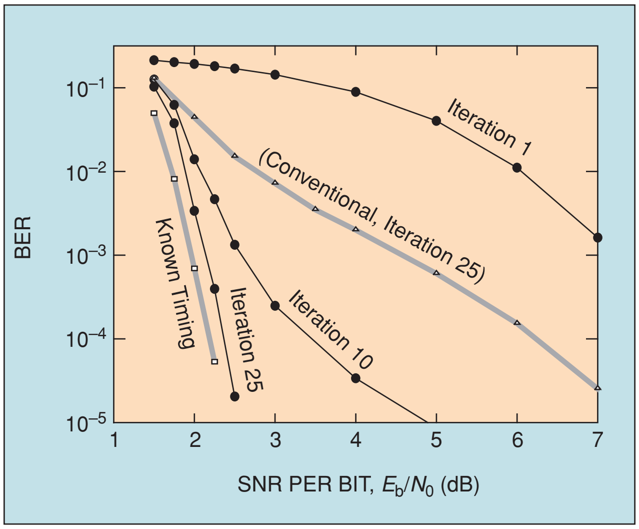
Consider the system of Figure 3 with a rate-1/4 recursive systematic convolutional encoder with generator $[1, 1, 1, (1 \oplus D)/(1 \oplus D \oplus D^2)]$, which maps blocks of 1278 message bits onto blocks of 5120 encoded bits. Assume the interleaver is spread random with parameter 16 and length 5120. The precoder is $1/(1 \oplus D^2)$, which is matched to the PR4 response. The timing model is a random walk

$$\tau_{k+1} = \tau_k + \sigma_w u_k, \qquad (7)$$

where $\{u_k\}$ are i.i.d. zero-mean, unit-variance Gaussian random variables, and where σ_w controls the severity of the random walk. To implement an assumption of perfect acquisition, we initialize with $\tau_0 = 0$. We consider a PLL-based estimator, which operates according to (2) and (3) but uses soft estimates of $b_k = a_k - a_{k-2}$ from the latest turbo-equalization iteration in place of a_k in (3), [4]. The resulting iterative receiver is shown in Figure 5. The results are shown in Figure 6, assuming $\sigma_w/T = 0.3\%$. Observe that the iterative receiver outperforms a conventional receiver (with timing recovery performed separately from turbo equalization) by 4.7 dB at BER = 2×10^{-5}, falling only 0.2 dB short of a genie-aided turbo equalizer that has perfect knowledge of the timing offsets ($\hat{\tau} = \tau$).



▲ *5. A method for jointly performing timing recovery, equalization, and decoding. This is essentially a traditional turbo equalizer but with an extra PLL and interpolation step added to each iteration. The modified turbo equalizer is able to compensate for a sloppy front-end PLL. The complexity of the PLL and interpolation functions are generally negligible compared to the equalizer and decoder, so that the overall system is only marginally more complex than a conventional turbo equalizer.*



▲ *6. Performance of the iterative receiver shown in Figure 5, with $\alpha = 0.025$, $\beta = 0$, a spread parameter of 16, and 21 interpolation coefficients. The channel was PR4 with $\sigma_w/T = 0.3\%$ and perfect acquisition.*

Figure 7 shows the timing waveforms for a sample packet at $E_b/N_0 = 4$ dB, where the severity of the random walk jitter was increased to $\sigma_w/T = 1\%$. The thick gray curve shows the actual $\{\tau_k\}$ sequence, plotted as a function of time k. The actual value of τ_k is near zero at time $k = 800$, but drops quickly so that it is near $-T$ at time $k = 1600$. The thin curve labeled 0 shows $\hat{\tau}_k$ after the front-end PLL; it is not able to track the rapid changes in τ_k. Rather, it hovers near zero until it eventually converges to approximately $\tau_k + T$, which is itself a stable but undesirable operating point. This phenomenon is called a cycle slip and is a common mode of failure for timing recovery systems.

A benefit of iterative timing recovery is that it automatically corrects for cycle slips. The curves in Figure 7 labeled 10, 20, etc. show $\hat{\tau}_k$ after the 10, 20, etc. iterations of the iterative receiver of Figure 5. Slowly but surely, from left to right, the iterative receiver "pinches" the $\hat{\tau}_k$ curve to match the τ_k curve. Eventually, after 95 iterations in this example, the $\hat{\tau}_k$ curve converges closely to the τ_k curve, meaning that the cycle slip has been corrected. In practice we can reduce the number of required iterations by detecting the cycle slip early on and correcting for it [5], [16], but we do not elaborate on these methods here.

Joint Soft Timing Recovery and Equalization

A different method for doing iterative timing recovery is to implement the scenario in Figure 1(d). Here we describe only the block labeled "joint timing recovery and equalization." Iterating between it on one side and the decoder on the other is then a simple extension of the turbo equalization methods presented in [10].

Basic Notation

Throughout this section, we will use discrete-time sequences. A discrete time sequence $\{a\}$ is a collection of variables $\ldots, a_{-2}, a_{-1}, a_0, a_1, a_2, \ldots$. A subsequence from time index i to time index j will be denoted by $a_i^j = [a_i, a_{i+1}, \ldots, a_j]$ if $j \geq i$. Otherwise, if $j < i$, a_i^j is the empty set $a_i^j = \emptyset$.

Quantized Timing Error Model

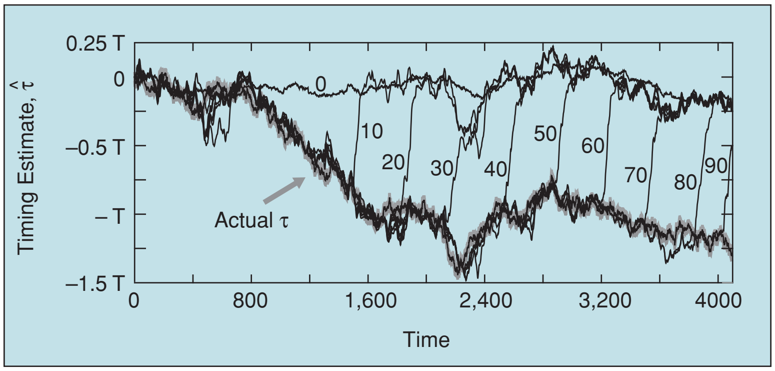
Consider the sampler in Figure 2. The signal $r(t)$ is sampled using a standard PLL sampler. We make an assumption that the sampled signal is of the form

$$r_k = \sum_{\ell} a_{k-\ell} h(\ell T - \varepsilon_k) + n_k, \qquad (8)$$

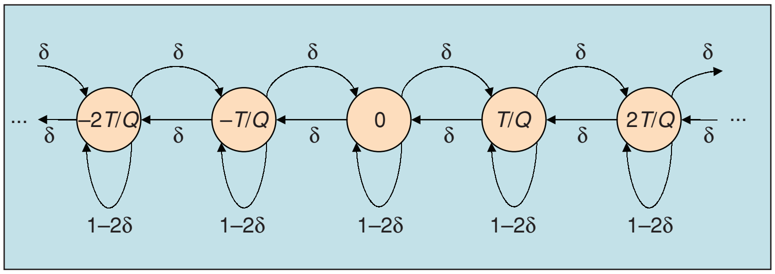
where ε_k is the timing error. (This can actually be derived from (1) under the assumption that τ_k and $\hat{\tau}_k$ are both slowly time varying.) Obviously, an accurate model would consider ε_k to be a real-valued random process. Without much loss in accuracy, however, we can assume that ε_k can take one of many values iT/Q, where i is an arbitrary integer and Q is a fixed positive integer

$$\varepsilon_k \in \mathcal{F} = \left\{ \ldots, \frac{-2T}{Q}, \frac{-T}{Q}, 0, \frac{T}{Q}, \frac{2T}{Q}, \ldots \right\}. \qquad (9)$$

Clearly, we have segmented the symbol interval of length T into Q segments. If Q is large, we do not lose much in terms of accuracy by performing this segmentation.



▲ 7. *An illustration of iterative timing recovery correcting a cycle slip. The thick gray curve shows the actual $\{\tau_k\}$ curve, plotted versus time k, whereas the thin black curves labeled 10, 20, etc., show the estimates $\{\hat{\tau}_k\}$ after iteration 10, 20, etc. These results are for the (4095, 3636) rate-8/9 code derived by puncturing the encoder* $[1, (1 \oplus D \oplus D^3 \oplus D^4)/(1 \oplus D \oplus D^4)]$, *assuming* $E_b/N_0 = 4$ *dB and* $\sigma_w/T = 1\%$.



▲ 8. *Markov chain for the timing error process* ε_k.

Markov Timing Error

Let us further assume that ε_k is a Markov chain. Obviously, this assumption cannot be justified by either (8) or assumption (9). It is intuitively clear, however, that if Q is sufficiently large and if the Markov chain memory is also large, we can approximate any discrete-time random process by a Markov chain of sufficiently large memory. Here, for the simplicity of the presentation, we will assume that the memory length of the Markov chain is one. Considering a larger-memory Markov chain would be a straightforward extension, so we do not pursue it here. Further, we assume that the Markov chain is governed by the following (known) transition probabilities

$$P(\varepsilon_k | \varepsilon_{k-1}) = \begin{cases} \delta & \text{if } \varepsilon_k = \varepsilon_{k-1} + T/Q \\ \delta & \text{if } \varepsilon_k = \varepsilon_{k-1} - T/Q \\ 1 - 2\delta & \text{if } \varepsilon_k = \varepsilon_{k-1} \\ 0 & \text{otherwise.} \end{cases} \qquad (10)$$

One can easily extend this model to include different ε_{k-1}-dependent probabilities for $\varepsilon_k = \varepsilon_{k-1} \pm iT/Q$ for $i > 1$, but for the simplicity of the presentation we do not consider such a case.

The states in the Markov chain in (10) are the values that the process ε_k can take, i.e., the states are in the set $\mathcal{F}$ defined in (9). At each time instant, we can increment the state by T/Q with probability δ, decrement the state by T/Q with probability δ, or remain in the same state with probability $1 - 2\delta$, as depicted in Figure 8. A sample realization of the random process ε_k is depicted in Figure 9.

The Problem Formulation

Assume that we send N binary independent and identically distributed, equally likely symbols $a_k \in \{-1, 1\}$ through the channel, where $1 \leq k \leq N$. So, $\Pr(a_k = 1) = 0.5$, for $1 \leq k \leq N$. Then we are faced with the following problem: We assume that the initial timing error is $\varepsilon_0 = 0$ and that the symbols up to time 0 are $a_k = -1$ for all $k \leq 0$. By assumption the signal model is (8) and the timing error model is (10). We sample the time axis $(0, NT]$ to collect samples $r_1^L = [r_1, r_2, \ldots, r_L]$. Note that generally $L \neq N$ because the timing error may accumulate over time, so we will not have the same number of samples as the number of symbols transmitted. At the receiver site, we obtain the symbols r_1^L and wish to determine the posterior probability of the kth symbol being either -1 or 1, i.e., we are interested in obtaining

$$\Pr\left(a_k = 1|r_1^L\right) \text{ and}$$
$$\Pr\left(a_k = -1|r_1^L\right) = 1 - \Pr\left(a_k = 1|r_1^L\right). \qquad (11)$$

The soft values in (11) are what we need to send to the decoder for iterative soft decoding.

Note that we are typically not interested in estimating the timing error ε_k because we are usually just interested in knowing the transmitted symbols a_k. Given the nature of the problem at hand and, given the solution that we are going to present, it will become clear that we will also be able to make an estimate of the timing error ε_k. We therefore can ask to compute the posterior probability of the timing error equaling any value $\psi \in \mathcal{F}$, where $\mathcal{F}$ is given in (9), i.e., we can seek to compute the soft timing error outputs

$$\Pr\left(\varepsilon_k = \psi|r_1^L\right) \quad \text{for all } \psi \in \mathcal{F}. \qquad (12)$$

Subsequently, we may seek to make the (hard) maximum a posteriori estimate of ε_k as

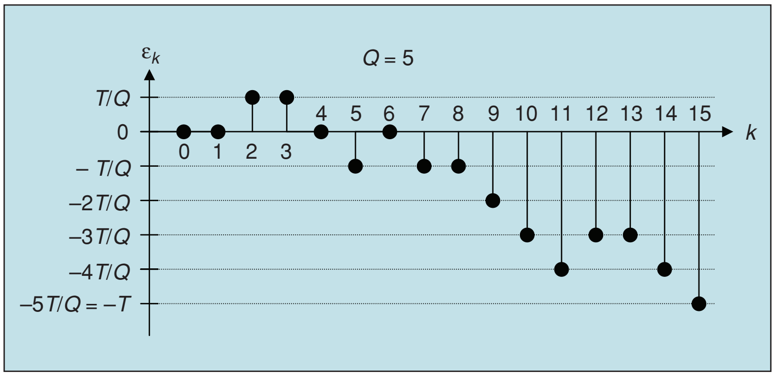
$$\hat{\varepsilon}_k = \arg\max_{\psi \in \mathcal{F}} \Pr\left(\varepsilon_k = \psi|r_1^L\right). \qquad (13)$$

The Timing Trellis

A problem with the Markov model in Figure 8 is that it has infinitely many states and thus does not permit trellis-based signal processing methods. For this reason, we next give an equivalent description with finitely many states which permits the formulation of a trellis.

Consider the sample realization of the process ε_k depicted in Figure 9 for $Q = 5$. Assume that we sample the time axis at time instants $kT - \varepsilon_k$. The exact sampling points are depicted by bullets (•) on the time axis t in Figure 10(a). Note that because we assumed that the timing error is quantized to multiples of (T/Q) the bullets must fall on integer multiples of $(T/Q) = (T/5)$, where T is the symbol duration.

Now, partition the time axis t into nonoverlapping semi-open intervals $\left((k-1)T, kT\right]$, where the length of each interval is the symbol duration T. Each interval has $Q = 5$ tick marks that denote possible positions (in multiples of (T/Q)), where a sample can be taken. From Figure 10(a), we notice that some intervals are sampled once, some intervals (in particular the sixth interval) are sampled twice, while some intervals (in particular, the fifth and seventh intervals) are not sampled at all.



▲ *9. Example: A sample of the realization of the timing error process ε_k.*

We define a new finite-state Markov process as follows. Denote the state of the process as q_k. The state is associated with a time interval $\left((k-1)T, kT\right]$, and can take one of the values in the following set:

$$q_k \in \mathcal{T} = \left\{0, 1_1, 2_2, \ldots, 1_Q, 2\right\}. \qquad (14)$$

The interpretation of the sampling-states q_k in the set $\mathcal{T}$ is as follows.

▲ State $q_k = 0 \in \mathcal{T}$ denotes that the kth symbol interval $\left((k-1)T, kT\right]$ is *not* sampled at all. For example, in Figure 10, the fifth and seventh symbol intervals have no samples. Therefore, we have $q_5 = 0$ and $q_7 = 0$.

▲ State $q_k = 1_i \in \mathcal{T}$, for $1 \le i \le Q$ denotes that the kth symbol interval $\left((k-1)T, kT\right]$ is sampled only once at the ith tick from the beginning of the interval. For example, in Figure 10, the third interval is sampled at the fourth tick from the start of the interval, so $q_3 = 1_4$. Similarly, the eighth interval is sampled at the first tick from the start of the interval, hence $q_8 = 1_1$.

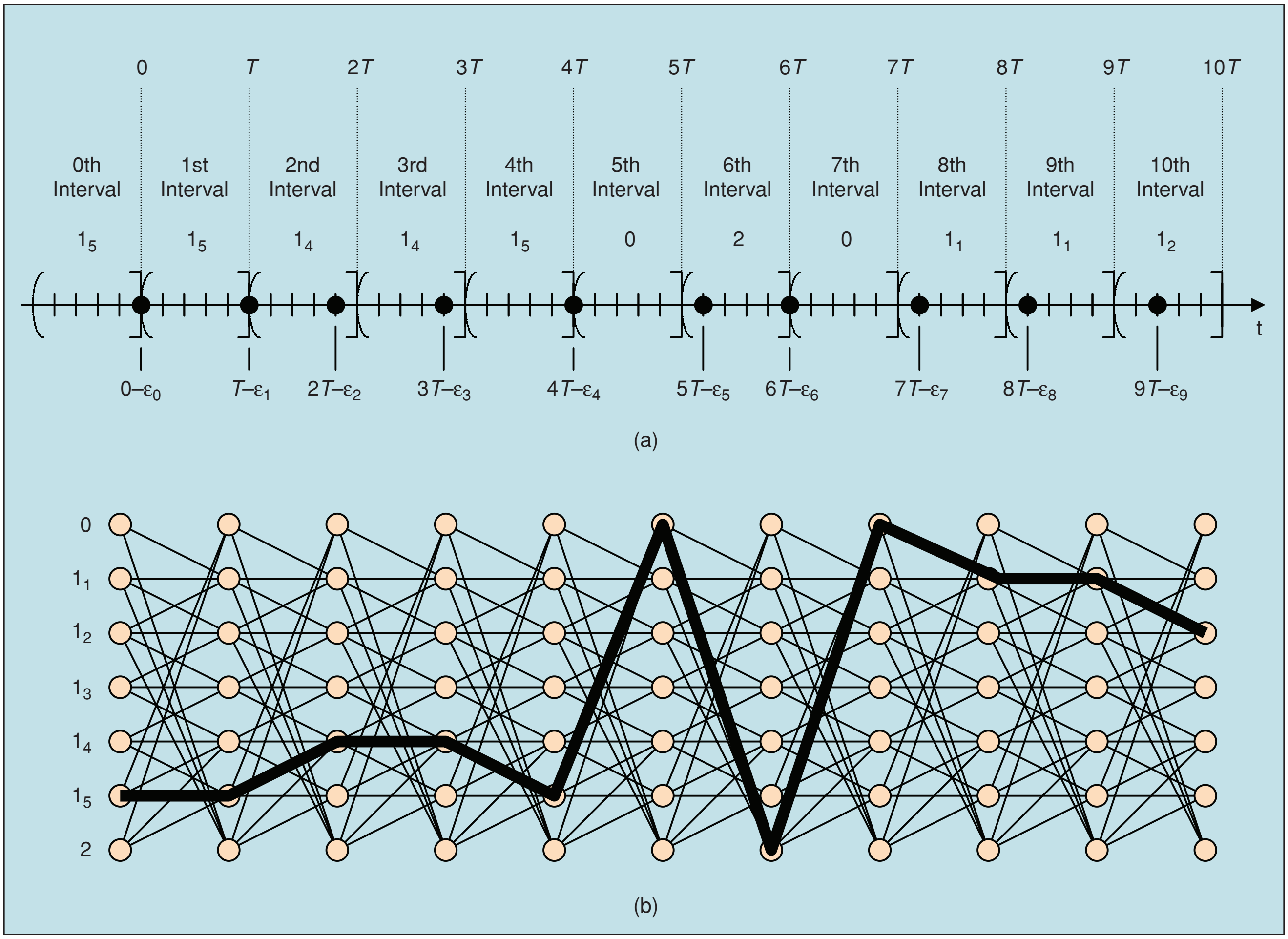
▲ State $q_k = 2 \in \mathcal{T}$ denotes that the kth symbol interval $\left((k-1)T, kT\right]$ is sampled twice. The only way an interval can be sampled twice is if it is sampled at the first and Qth ticks. The constraints of the Markov process in (10) prevent any other way of two samples falling in the same interval. In Figure 10, the sixth interval is sampled twice, hence $q_6 = 2$.

Note that in Figure 10, the sequence of states $1_5, 1_5, 1_4, 1_4, 1_5, 0, 2, 0, 1_1, 1_1, 1_2, \ldots$, uniquely determines where the time axis is sampled. In fact, there is a one-to-one correspondence between the timing error sequence ε_k and the state sequence q_k.

Obviously, given assumption (10), there are some restrictions on how the sampling-states q_k can evolve. To represent all valid sampling-state transitions, we form a timing trellis, depicted in Figure 11. To a branch in the timing trellis, we associate a transition probability $\Pr(q_k|q_{k-1})$. The transition probabilities can be computed from the Markov assumption (10). For $Q = 5$, the transition probabilities derived from (10) are listed in Figure 11.

Key Feature of the Timing Trellis

The key feature of the timing trellis in Figure 11 is that the branches in the trellis carry a variable number of samples. We will denote the vector of samples taken in the timing interval $(k-1)T, kT]$ by z_k. Note that z_k could be an empty vector if no sample is taken in the kth symbol interval. Referring to Figure 10, we see that the first interval is



▲ *10. (a) Time axis sampled at time instants corresponding to the timing error realization in Figure 8. (b) The timing trellis, where the thick path corresponds to the sampling realization depicted in the top part of the figure.*

sampled once and, therefore, $z_1 = r(T - \varepsilon_1) = r_1$. However, the fifth symbol interval is not sampled at all, so z_5 is an empty vector $z_5 = \emptyset$. On the other hand, the sixth interval is sampled twice, so $z_6 = [r(5T - \varepsilon_5), r(6T - \varepsilon_6)] = [r_5, r_6]$.

The Joint ISI-Timing Trellis

Denote by I the ISI length of the pulse $h(t)$ in (8). For example, consider the following pulse depicted in Figure 12(a)

$$h(t) = \begin{cases} \frac{4}{3}\sin^2\left(\frac{2\pi(t+T)}{6T}\right), & \text{for } -T \le t \le 2T \\ 0, & \text{otherwise.} \end{cases} \qquad (15)$$

The pulse in Figure 12(a) spans three symbol intervals, so its ISI length is $I = 2$.

Let us consider first the case when $\varepsilon_k = 0$ for all $k > 0$, i.e., the case when there is no timing error. Then the pulse $h(t)$ is sampled at integer multiples of T. We conclude that the only nonzero samples of $h(t)$ are then $h(T) = 1$ and $h(2T) = 1$, both denoted by circles in Figure 12(a). Consequently, we may rewrite (8) for the signal sample as

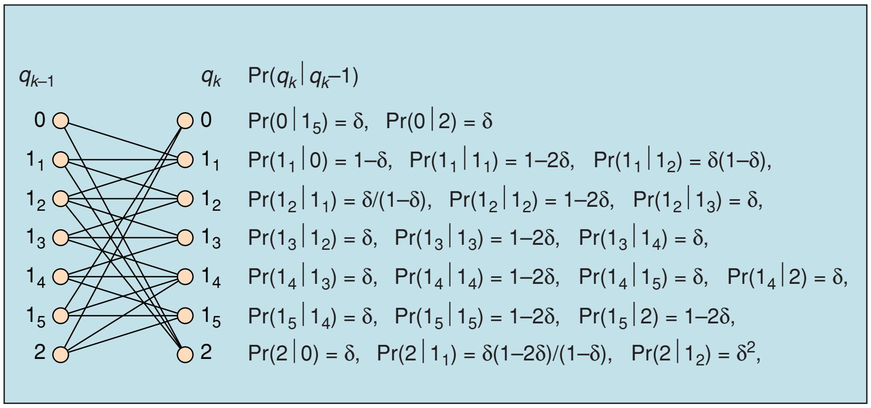
$$r_k = a_k + a_{k-1} + n_k. \qquad (16)$$

As the next example, consider a constant nonzero timing error, say $\varepsilon_k = 2T/Q = 2T/5$, for all $k > 0$. Then, the signal $h(t)$ is sampled at time instants $kT - 2T/5$. In Figure 12(a), these samples (denotes by stars) are $h(-2T/5) = 0.461$, $h(3T/5) = 1.319$, and $h(8T/5) = 0.221$. Consequently, we may rewrite (8) as

$$r_k = 0.461 \cdot a_k + 1.319 \cdot a_{k-1} + 0.221 \cdot a_{k-2} + n_k. \qquad (17)$$

Equations (16) and (17) illustrate that the value of the timing error ε_k influences the shape of the ISI, since the two equations are obviously different. Therefore, as the timing error ε_k changes its value with k, so does the nature of the ISI. We next need to combine the timing trellis (depicted in Figure 11) with a proper model of ISI. The result is another trellis, which is a bit more complicated because it captures, jointly, the ISI and timing error.

First, consider a trellis that purely captures the transitions from a transmitted symbol subsequence $(a_{k-I}, a_{k-I+1}, \ldots, a_{k-1})$ to $(a_{k-I+1}, a_{k-I+2}, \ldots, a_k)$. If we define the set $\mathcal{I} = \{-1, 1\}^I$, then clearly


q_{k-1}
q_k
$\Pr(q_k | q_k - 1)$
0
1_1
1_2
1_3
1_4
1_5
2
0
1_1
1_2
1_3
1_4
1_5
2
$\Pr(0|1_5) = \delta, \quad \Pr(0|2) = \delta$
$\Pr(1_1|0) = 1-\delta, \quad \Pr(1_1|1_1) = 1-2\delta, \quad \Pr(1_1|1_2) = \delta(1-\delta),$
$\Pr(1_2|1_1) = \delta/(1-\delta), \quad \Pr(1_2|1_2) = 1-2\delta, \quad \Pr(1_2|1_3) = \delta,$
$\Pr(1_3|1_2) = \delta, \quad \Pr(1_3|1_3) = 1-2\delta, \quad \Pr(1_3|1_4) = \delta,$
$\Pr(1_4|1_3) = \delta, \quad \Pr(1_4|1_4) = 1-2\delta, \quad \Pr(1_4|1_5) = \delta, \quad \Pr(1_4|2) = \delta,$
$\Pr(1_5|1_4) = \delta, \quad \Pr(1_5|1_5) = 1-2\delta, \quad \Pr(1_5|2) = 1-2\delta,$
$\Pr(2|0) = \delta, \quad \Pr(2|1_1) = \delta(1-2\delta)/(1-\delta), \quad \Pr(2|1_2) = \delta^2,$

▲ *11. A trellis section of the timing trellis and the corresponding transition probabilities.*

$(a_{k-I+1}, a_{k-I+2}, \ldots, a_k) \in \mathcal{I}$. For $I = 2$, an example of this trellis is given in Figure 12(b). The trellis in Figure 12(b) is a standard trellis used for detection of symbols over an ISI channel with perfect timing (see [10] for details).

Next, we define the joint ISI-timing trellis by merging the ISI trellis in Figure 12(b) and the timing trellis in Figure 11. A state $s_k \in \mathcal{S}$ at time instant k of the joint ISI-timing trellis is determined by a pair of states; the first state coming from the ISI trellis $\mathcal{I}$, i.e., $(a_{k-I+1}, a_{k-I+2}, \ldots, a_k)$, and the second coming from the timing trellis $\mathcal{T}$, i.e., $q_k \in \mathcal{T}$:

$$s_k = \left(a_{k-I+1}, a_{k-I+2}, \ldots, a_k, q_k\right) \in \mathcal{S} = \mathcal{I} \times \mathcal{T}. \tag{18}$$

Since the timing trellis has $Q + 2$ states and since the ISI trellis has 2^I states, the joint ISI-timing trellis has $(Q + 2)2^I$ states. An example of a joint ISI-timing trellis is given in Figure 12(c), which is a result of merging the trellises in Figures 12(b) and 11. Note that the joint ISI-timing trellis in Figure 12(c) has $(Q + 2) \cdot 2^I = (5 + 2) \cdot 2^2 = 28$ states, which is why we only give a subset of the states. A branch in the joint ISI-timing trellis Figure 12(c) exists if and only if a corresponding branch exists in *both* the ISI and timing trellises of Figures 12(b) and 11.

Note that the joint ISI-timing trellis inherits the key feature of the timing trellis; namely, the number of samples carried by a branch of the joint ISI-timing trellis can be either zero, one, or two.

A BCJR-Like Algorithm

The joint ISI-timing trellis admits the execution of the sum-product algorithm presented in [17]. When the timing error is $\varepsilon_k = 0$ for all $k > 0$, this algorithm takes the form of the Bah–Jelinek–Cocke–Raviv (BCJR) algorithm, which is a standard component of iterative turbo equalization [10]. In principle, we could end the discussion here and simply state that the application of the sum-product algorithm [17] on the joint ISI-timing trellis in Figure 12(c) solves the problem stated earlier. Since the joint ISI-timing trellis has the unique feature that different branches carry different numbers of samples, however, a detailed presentation of the BCJR-like sum-product algorithm for the joint ISI-timing trellis is in order. We will not derive the sum-product message-passing formulas but rather, just state them, since they are very similar to the ones in [11] (see also [10] and [17] in this issue). The reader is advised to first read [11] before proceeding.

Definitions

First, we note that the symbols a_k are i.i.d. with probability 0.5 and we use the Markov property of the timing trellis in Figure 11 to derive the conditional probability of reaching s_k from s_{k-1}

$$\begin{aligned} \Pr\left(s_k | s_{k-1}, s_{k-2}, \ldots\right) &= \Pr\left(s_k | s_{k-1}\right) \\ &= \Pr(a_k) \Pr\left(q_k | q_{k-1}\right) \\ &= \frac{1}{2} \Pr(q_k | q_{k-1}). \end{aligned} \tag{19}$$

We next define the following variables that are needed to execute the BCJR-like sum-product algorithm on the joint ISI-timing trellis.

▲The α-coefficient $\alpha(k, m, i)$ is defined as the probability that the kth ISI-timing state s_k equals $m \in \mathcal{S}$ and that there were i samples during the first k symbol intervals

$$\alpha(k, m, i) = \Pr\left(s_k = m, z_1^k = r_1^i\right). \tag{20}$$

▲The β-coefficient $\beta(k, t, i)$ is defined as the conditional probability that there were $L - i$ samples in the last $N - k$ symbol intervals, given that the kth ISI-timing state s_k equals $m \in \mathcal{S}$

$$\beta(k, m, i) = \Pr\left(z_{k+1}^N = r_{i+1}^L | s_k = m\right). \tag{21}$$

▲The γ-coefficient is defined as follows. Let $s_{k-1} = m' \in \mathcal{S}$ be the state at time $k - 1$ and let $s_k = m \in \mathcal{S}$ be the state at time k. Then the γ-coefficient for the branch (m', m) is defined as the joint probability of the state at time k and the sample vector z_k at time k, given the knowledge of the state at time $k - 1$

$$\gamma(k, m', m, i) = \begin{cases} P(s_k = m, z_k = \emptyset | s_{k-1} = m') & \text{if } m \in \mathcal{I} \times \{0\} \\ P(s_k = m, z_k = r_i | s_{k-1} = m') & \text{if } m \in \mathcal{I} \times \\ & \quad \left\{1_1, 1_2, \ldots, 1_Q\right\} \\ P(s_k = m, z_k = r_{i-1}^i | s_{k-1} = m') & \text{if } m \in \mathcal{I} \times \{2\}. \end{cases} \tag{22}$$

Notice that the definition of the γ-coefficient differs with the value of the timing state q_k to compensate of the different number of samples carried by different branches of the joint ISI-timing trellis.

Let the state at time $k-1$ be $s_{k-1} = m' = (a_{k-2}, a_{k-1}, q_{k-1}) = (-1, -1, 1_2)$, for example, and let the state at time k be $s_k = m = (a_{k-1}, a_k, q_k) = (-1, 1, 1_2)$. First notice that if $(a_{k-2}, a_{k-1}, a_k) = (-1, -1, 1)$ and if $q_k = 1_2$, we get $r_i = 0.461 \cdot (-1) + 1.319 \cdot (-1) + 0.221 \cdot 1 + n_i \quad = -1.559 + n_i$ from (17). Assuming that the noise in the channel is white Gaussian with variance σ^2, the γ-coefficient for the branch (m', m) is given by

$$\begin{aligned}\gamma(k, m', m, i) &= P\left(s_k = m, z_k = r_i | s_{k-1} = m'\right)\\ &= P\left(z_k = r_i | s_{k-1} = m', s_k = m\right)\\ &\quad \times P\left(s_k = m | s_{k-1} = m'\right)\\ &= \frac{\exp\left[-\frac{(r_i+1.559)^2}{2\sigma^2}\right]}{\sqrt{2\pi\sigma^2}} P\left(a_k = 1\right)\\ &\quad \times \Pr\left(q_k = 1_2 | q_{k-1} = 1_2\right)\\ &= \frac{\exp\left[-\frac{(r_i+1.559)^2}{2\sigma^2}\right]}{\sqrt{2\pi\sigma^2}} \cdot \frac{1}{2} \cdot \delta. \end{aligned} \tag{23}$$

Similarly, by properly accounting for the ISI and by utilizing the transition probabilities in Figure 11, we can obtain all the other γ-coefficients for all other branches in the joint ISI-timing trellis.

With this notation, we can formulate a BCJR-like algorithm for performing joint soft timing recovery and symbol detection.

Recursions

Using the Markov properties of the joint ISI-timing process, it can be shown [18] that the α and β coefficients satisfy the following recursive forward and backward relations, respectively

$$\alpha(k, m, i) = \begin{cases} \sum\limits_{m' \in \mathcal{S}} \alpha(k-1, m', i-1) \cdot \gamma(k, m', m, i) \\ \quad \text{if } m \in \mathcal{I} \times \left\{1_1, 1_2, \ldots, 1_Q\right\} \\ \sum\limits_{m' \in \mathcal{S}} \alpha(k-1, m', i) \cdot \gamma(k, m', m, i) \\ \quad \text{if } m \in \mathcal{I} \times \{0\} \\ \sum\limits_{m' \in \mathcal{S}} \alpha(k-1, m', i-2) \cdot \gamma(k, m', m, i) \\ \quad \text{if } m \in \mathcal{I} \times \{2\} \end{cases} \tag{24}$$

The gist of iterative timing recovery is simple: rather than performing timing recovery once, timing recovery is performed multiple times while interacting with the decoder.

$$\begin{aligned}\beta(t, m, i) = &\sum_{m' \in \mathcal{I} \times \left\{1_1, 1_2, \ldots, 1_Q\right\}} \beta(k+1, m', i+1)\\ &\cdot \gamma(k+1, m, m', i+1)\\ &+ \sum_{m' \in \mathcal{I} \times \{0\}} \beta(k+1, m', i)\\ &\cdot \gamma(k+1, m, m', i)\\ &+ \sum_{m' \in \mathcal{I} \times \{2\}} \beta(k+1, m', i+2)\\ &\cdot \gamma(k+1, m, m', i+2). \end{aligned} \tag{25}$$

Initial Conditions

The forward (24) and backward (25) recursions need to be started with a set of initial coefficients $\alpha(0, m, i)$ and $\beta(N, m, i)$. These initial coefficients should be chosen such that they reflect some prior knowledge of the timing error and transmitted symbols at the beginning and the end of the transmitted block. If we know that $a_k = -1$ for $k \le 0$, we can set

$$\alpha(0, m, i) = \begin{cases} 1 & \text{if } \ i = 0 \ \text{ and } \ m = \left(-1, -1, 1_Q\right) \\ 0 & \text{otherwise} \end{cases}$$

$$\beta(N, m, i) = \begin{cases} 1 & \text{if } \ i = L \ \text{ and all; } m \in \mathcal{S} \\ 0 & \text{otherwise.} \end{cases}$$

Since the influence of initial conditions dies exponentially as we move away from the boundaries, even if we choose improper boundary conditions, the impact will not propagate far.

Soft Symbol Estimates

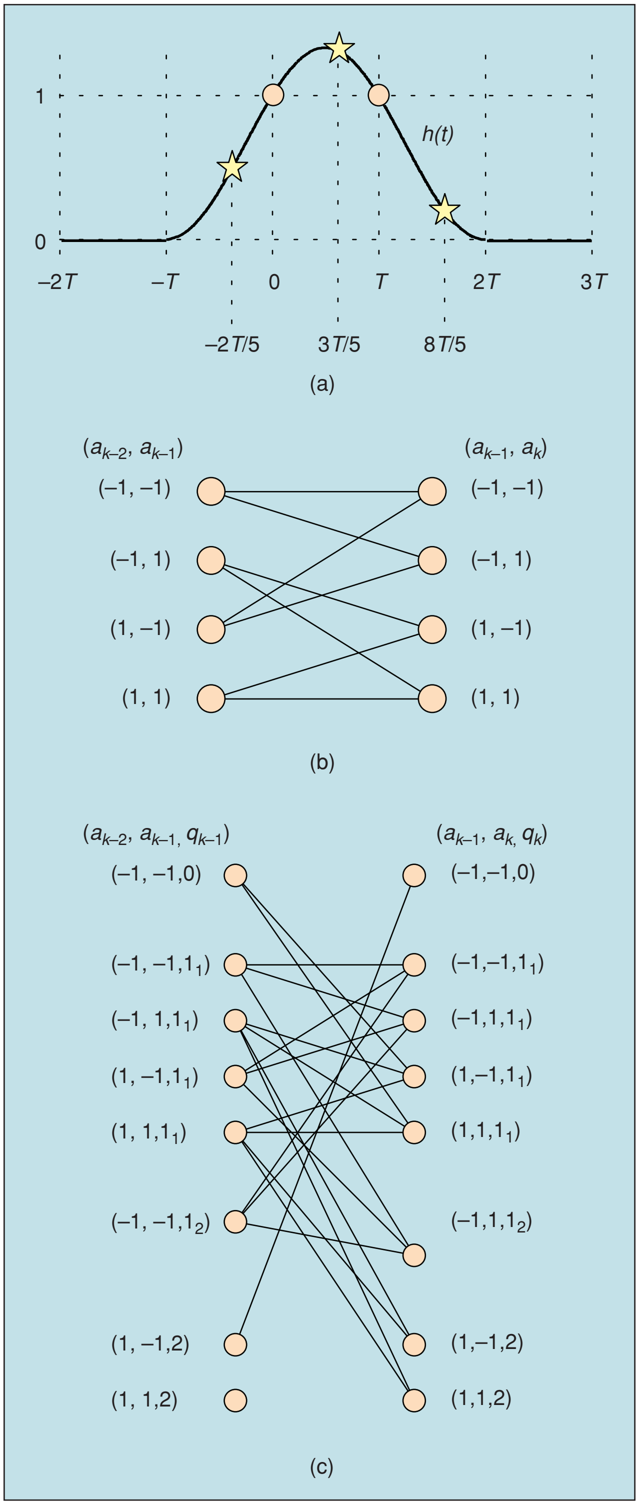
After completing the forward-backward recursions (24) and (25), we are in the position to compute the posterior probabilities of the joint-states $s_k = m \in \mathcal{I} \times \mathcal{T}$

$$\begin{aligned}\lambda_k(m) &= \Pr\left(s_k = m, z_1^N = r_1^L\right)\\ &= \sum_{i=1}^{L} \alpha(k, m, i)\beta(k, m, i). \end{aligned} \tag{26}$$

The soft output, which is given as the ratio of posterior probabilities of two different symbol values, can then be computed as

$$\frac{\Pr\left(a_k=1|r_1^L\right)}{\Pr\left(a_k=-1|r_1^L\right)}=\frac{\sum_{m:a_k=1}\lambda_k(m)}{\sum_{m:a_k=-1}\lambda_k(m)}. \tag{27}$$

The value in (27) is the quantity that is passed to the decoder in Figure 1(d) for a round of iterative decoding. The decoder, after performing its task, passes a new value of the posterior probability, which is to be used as the new prior probability instead of $P(a_k)$ in (19). This completes an iterative cycle.



▲ *12. (a) An example of a pulse with intersymbol interference length $I=2$ that spans $I+1=3$ symbol intervals. (b) A trellis that captures the ISI behavior, i.e., that captures the transition rules from a symbol subsequence (a_{k-2}, a_{k-1}) to (a_{k-1}, a_k). (c) A joint trellis that captures both the timing error and the ISI behavior.*

Soft Timing Estimates

The a posteriori probability of the timing error for any value $\psi\in\mathcal{F}$ can be computed as in [18]

$$\begin{aligned}\Pr\left(\varepsilon_i=\psi\mid r_1^L\right)=&\\ &\sum_{k,j:(i-k+1)T+\frac{j}{Q}T=\psi}\left[\sum_{m:m\in\mathcal{I}\times\{1_j\}}\alpha(k,m,i)\beta(k,m,i)\right]\\ &+\sum_{k:(i-k)T=\psi}\left[\sum_{m:m\in\mathcal{I}\times\{2\}}\alpha(k,m,i)\beta(k,m,i)\right]\\ &+\sum_{k:(i-k+1)T+\frac{T}{Q}=\psi}\left[\sum_{m:m\in\mathcal{I}\times\{2\}}\alpha(k,m,i+1)\right.\\ &\qquad\left.\cdot\beta(k,m,i+1)\right],\end{aligned} \tag{28}$$

for all integers $1\le i\le L$. Given the formula for $\Pr\left(\varepsilon_k=\psi\mid r_1^L\right)$ in (28), we can easily find the maximum a posteriori timing error estimate using (13).

Some Implementation Issues

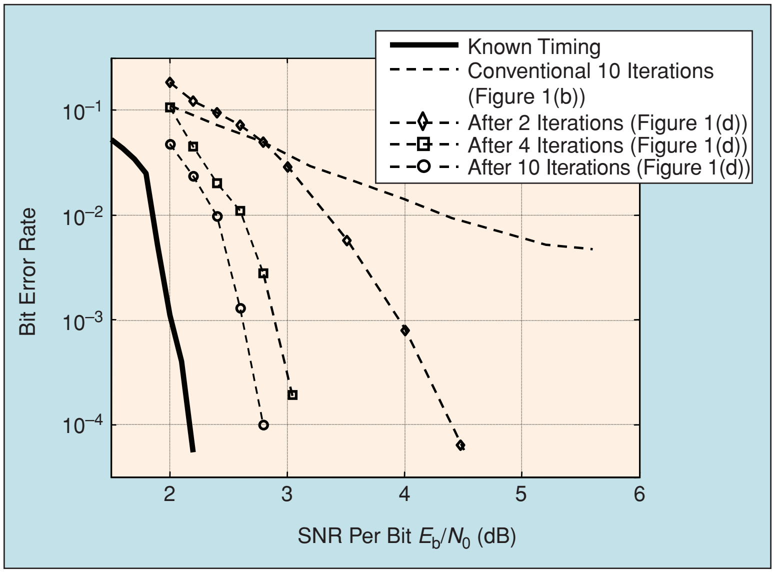
Just as in the classical BCJR algorithm [11], we need to normalize the α and β coefficients to prevent overflow errors. Another issue is that the computational complexity and memory requirements for the method are very high, proportional to the square of the block length. This can be vastly reduced by reverting to windowed techniques [18].

Simulation Results

To illustrate the applicability of the joint timing/equalization method, we replicate the simulations shown in Figures 6 and 7. The points of difference from Figures 6 and 7 are: 1) we employ the scenario of Figure 1(d) rather than Figure 1(c); 2) the joint timing recovery and equalization is performed using the algorithm presented in this section; 3) the codes are low-density parity-check (LDPC) codes rather than convolutional codes, but the code rates are kept the same; 4) a message-passing decoder of LDPC codes is used in place of a decoder of convolutional codes; and 5) no precoder is employed.

Figure 13 presents the same results as Figure 6, under the same channel and timing error assumption and for the same code rate (1/4). The chosen code was a regular LDPC code, whose variable-node degree coefficient is three. The actual timing error is a Gaussian random walk just as in Figure 6, but the receiver pretends that the timing error follows the first-order Markov process quantized to $Q = 5$ quantization levels and performs iterative detection using the scenario in Figure 1(d). A comparison of Figures 6 and 13 shows that the method in Figure 1(d) requires, on average, fewer visits to the timing/equalization block (less iterations) than the method in Figure 1(c). The price paid is the increased complexity of the joint timing/equalization unit.

Figure 14 shows the estimated timing error for a sample realization of a channel by using the scenario in Figure 1(d). The channel, timing error, and code rate (8/9) are assumed to be the same as in Figure 7. Again, the chosen code was a regular LDPC code whose variable-node degree is three. The actual timing error is a Gaussian random walk just as in Figure 7, but the receiver pretends that the timing error follows the first-order Markov process quantized to $Q = 10$ quantization levels and performs iterative detection using the scenario in Figure 1(d). Figure 14 shows that after the first iteration, there is clearly a cycle slip present (starting at time $k \approx 1300$). After the second iteration, the cycle slip is eliminated, but sporadic jumps in the timing error estimate still occur. After the third iteration, even the sporadic jumps are eliminated.

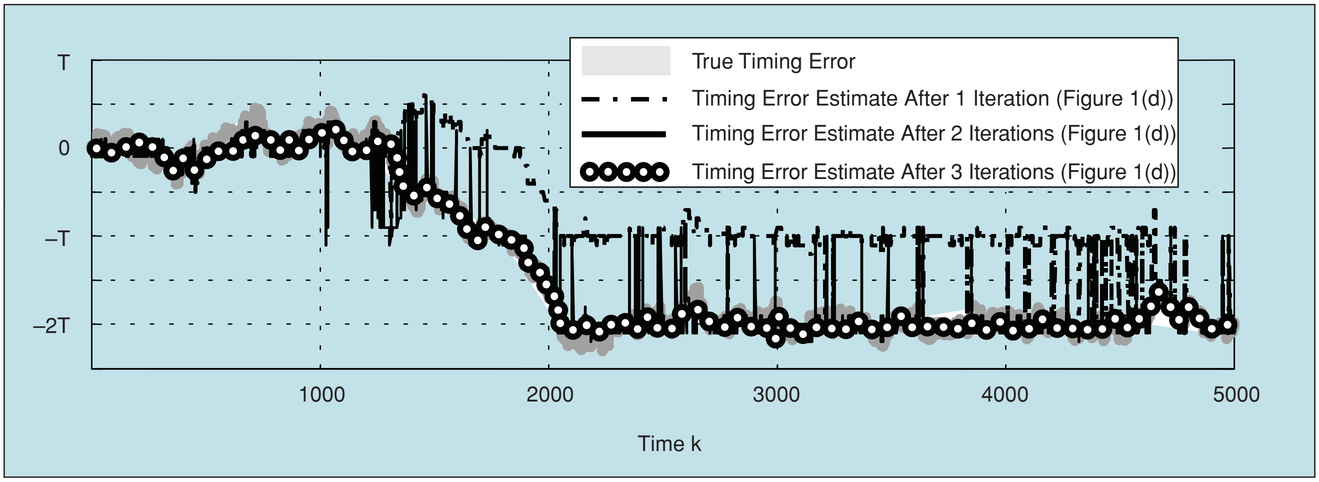


▲ *13. Bit error rate performance of the joint soft timing/equalization method [Figure 1(d)] in conjunction with an LDPC decoder. The channel and timing error parameters as well as the code rate are the same as in Figure 6.*

Conclusions

Conventional approaches to timing recovery fail at low SNR because they ignore error-control coding. We have described how iterative timing recovery exploits the power of error-control codes to enable reliable operation at very low SNR, with performance that closely matches a genie-aided receiver that has perfect timing knowledge. A receiver that uses iterative timing recovery is often only marginally more complex than a receiver that uses conventional timing recovery and either conventional turbo equalization or conventional turbo decoding or message-passing decoding.

John R. Barry received the B.S. degree in electrical engineering from the State University of New York at Buffalo in 1986 and the M.S. and Ph.D. degrees in electrical engineering from the University of California at Berkeley in 1987 and 1992, respectively. Since 1992 he has been with the Georgia Institute of Technology,



▲ *14. True and estimated timing error when employing the scenario of Figure 1(d) with an LDPC decoder. The channel and timing error parameters as well as the code rate are same as in Figure 7.*

Atlanta, where he is an associate professor with the School of Electrical and Computer Engineering. His research interests include wireless communications, equalization, and multiuser communications. He is the author of *Wireless Infrared Communications* (Kluwer, 1994) and a coauthor of *Digital Communications,* 3rd ed. (Kluwer, 2004).

Aleksandar Kavčić received the Dipl.-Ing. degree in electrical engineering from Ruhr-University, Bochum, Germany, in 1993 and the Ph.D. degree in electrical and computer engineering from Carnegie Mellon University, Pittsburgh, Pennsylvania, in 1998. Since 1998, he has been with the Division of Engineering and Applied Sciences of Harvard University, where he is a John L. Loeb Associate Professor of Natural Sciences. From 1998 to 2002 he was an assistant professor of electrical engineering at Harvard University. His research spans topics in communications, signal processing, information theory, and magnetic recording, with most recent interests in magnetic recording channel modeling, multichannel signal processing, detector design, timing recovery, and iterative decoding algorithms. He received the IBM Partnership Award in 1999 and the NSF CAREER Award in 2000. He is on the editorial board of *IEEE Transactions on Information Theory.*

Steven W. McLaughlin received the B.S. degree from Northwestern University in 1985, the M.S. degree from Princeton University in 1986, and the Ph.D. degree from the University of Michigan in 1992, all in electrical engineering. He joined the School of Electrical and Computer Engineering at Georgia Tech in September 1996, where he is currently an associate professor. He is also a principal scientist at Calimetrics, Inc. He received the NSF CAREER Award and the Presidential Early Career Award for Scientists and Engineers (PECASE) in 1997, the International Storage Industries Consortium Technical Achievement Award in 2002, and the Friend of the Graduate Student Award in 2002 from the GT Graduate Student Association. He is currently second vice president of the IEEE Information Theory Society. He was publications editor for *IEEE Transactions on Information Theory.* He co-edited (with Sergio Verdu) *Information Theory: 50 Years of Discovery* (IEEE Press, 1999). His research interests are in the area of communications and information theory, specifically in turbo and constrained codes for magnetic and optical recording, FEC and equalization for optical networks, theory of error control coding, and source coding (quantization and compression).

Aravind Nayak received his B.Tech. degree in 1999 from the Indian Institute of Technology, Madras, and his M.S. in 2000 from the Georgia Institute of Technology, where he is currently a doctoral student. His research interests are in error correction codes and the use of iterative error correction codes to improve timing recovery. He is a student member of the IEEE.

Wei Zeng received his B.Eng. degree in 1999 from Shanghai JiaoTong University, China, and his M.S. in 2002 from the University of Hawaii, Manoa. He is currently a doctoral student at Harvard University. His research interests include detection and estimation, information theory, and signal processing in timing recovery. He is a student member of the IEEE.

References

[1] J. Hamkins and D. Divsalar, "Coupled receiver-decoders for low rate turbo codes," in *Proc. IEEE Int. Symp. Information Theory,* Yokohama, Japan, June 2003, p. 381.

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

[3] A.R. Nayak, J.R. Barry, and S.W. McLaughlin, "Lower bounds for the performance of iterative timing recovery at low SNR," in *Proc. 15th Int. Symp. Mathematical Theory Networks and Systems,* Notre Dame, IN, Aug. 2002, pp. 12–16.

[4] A.R. Nayak, J.R. Barry, and S.W. McLaughlin, "Joint timing recovery and turbo equalization for coded partial response channels," *IEEE Trans. Magn.,* vol. 38, pp. 2295–2297, Sept. 2002.

[5] X. Jin and A. Kavčic´, "Cycle-slip-detector-aided iterative timing recovery," *IEEE Trans. Magn.,* vol. 38, pp. 2292–2294, Sept. 2002.

[6] V. Lottici and M. Luise, "Carrier phase recovery for turbo-coded linear modulations," in *Proc. IEEE Int. Conf. Communications,* New York, Apr. 2002, pp. 1541–1545.

[7] A. Burr and L. Zhang, "Iterative joint synchronization and turbo-decoding," in *Proc. IEEE Int. Symp. Inform. Theory,* Lausanne, Switzerland, June 2002, p. 414.

[8] R. Nuriyev and A. Anastasopoulos, "Analysis of joint iterative decoding and phase estimation for the noncoherent AWGN Channel using density evolution," in *Proc. IEEE Int. Symp. Inform. Theory,* Lausanne, Switzerland, June 2002, p. 168.

[9] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes," in *Proc. IEEE Int. Conf. Commun.,* Geneva, Switzerland, May 1993, pp. 1064–1070.

[10] R. Kötter, A.C. Singer, and M. Tüchler, "Turbo equalization," *IEEE Signal Processing Mag.,* vol. 20, pp. 67–80, Jan. 2004.

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

[12] L.R. Rabiner and B.H. Juang, "An introduction to hidden Markov models," *IEEE Acoust., Speech, Signal Processing Mag.,* vol. 3, pp. 4–16, 1986.

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

[14] G. Vishakhadatta et al. "An EPR4 read/write channel with digital timing recovery," *IEEE J. Solid-State Circuits,* vol. 33, pp. 1851–1857, Nov. 1998.

[15] H. Meyr, M. Moeneclaey, and S.A. Fechtel, *Digital Communication Receivers.* New York: Wiley, 1997.

[16] X. Jin and A. Kavčic´, "Cycle-slip detection using soft-output information," in *Proc. IEEE Intern. Conf. Commun.,* vol. 9, 2001, pp. 2706–2710.

[17] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Processing Mag.,* vol. 20, pp. 28–41, Jan. 2004.

[18] W. Zeng and A. Kavčic´, "Optimal soft-output detector for channels with intersymbol interference and timing errors," *IEEE Trans. Magn.,* vol. 39, pp. 2555–2557, Sept. 2003.