2004

Synchronization for capacity -approaching coded communication systems

Jian Sun
West Virginia University

Synchronization for Capacity-Approaching Coded Communication Systems

Jian Sun

Dissertation submitted to the
College of Engineering and Mineral Resources
at West Virginia University
in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy
in
Electrical Engineering

Matthew C. Valenti, Ph.D., Chair
Erdogan Gunel, Ph.D.
Mark A. Jerabek, Ph.D.
Ronald L. Klein, Ph.D.
Daryl Reynolds, Ph.D.

Lane Department of Computer Science and Electrical Engineering
West Virginia University
Morgantown, West Virginia
2004

Keywords: Synchronization, Turbo Codes, LDPC Codes, Frame Synchronization

ABSTRACT

Synchronization for Capacity-Approaching Coded Communication Systems

Jian Sun

The dissertation concentrates on synchronization of capacity approaching error-correction codes that are deployed in noisy channels with very low signal-to-noise ratio (SNR). The major topics are symbol timing synchronization and frame synchronization.

Capacity-approaching error-correction codes, like turbo codes and low-density parity-check (LDPC) codes, are capable of reaching very low bit error rates and frame error rates in noisy channels by iterative decoding. To fully achieve the potential decoding capability of Turbo codes and LDPC codes, proper symbol timing synchronization, frame synchronization and channel state estimation are required. The dissertation proposes a joint estimator of symbol time delay and channel SNR for symbol timing recovery, and a maximum *a posteriori* (MAP) frame synchronizer for frame synchronization.

Symbol timing recovery is implemented by sampling and interpolation. The received signal is sampled multiple times per symbol period with unknown delay and unknown SNR. A joint estimator estimates the time delay and the SNR. The signal is rebuilt by interpolating available samples using estimated time delay. The intermediate decoding results enable decision-feedback estimation. The estimates of time delay and SNR are refined by iterative processing. This refinement improves the system performance significantly.

Usually the sampling rate is assumed to be a strict integer multiple of the symbol rate. However, in a practical system the local oscillators in the transmitter and the receiver may have random drifts. Therefore the sampling rate is no longer an exact multiple of the symbol rate, and the sampling time follows a random walk. This random walk may harm the system performance severely. The dissertation analyzes the effect of random time walks and proposes to mitigate the effect by overlapped sliding windows and iterative processing.

Frame synchronization is required to find the correct boundaries of codewords. MAP frame synchronization in the sense of minimizing the frame sync failure rate is investigated. The MAP frame synchronizer explores low-density parity-check attributes of the capacity-approaching codes. The accuracy of frame synchronization is adequate for considered coded systems to work reliably under very low SNR.

Acknowledgement

I would first like to thank my parents. They are both electrical engineers in the field of radar and are still providing me with valuable advices on the basis of their abundant experiences.

I am obliged to my advisor, Dr. Valenti, an excellent teacher, researcher, team leader and mentor. I have been enjoying working with him and learning from him. He gave me the opportunity to start working on this interesting topic besides other projects in wireless communications. He helped me prepare all my papers with very detailed and knowledgeable comments. He made me feel comfortable with English, in both writing and verbal communications.

Next I would like to thank all the committee members for taking their most valuable time to read the manuscripts of my preliminary report and dissertation. They gave very insightful reviews and informative comments to the manuscripts. Their hard and prompt working style encouraged me to prepare for graduation.

Finally, this dissertation is dedicated to my beloved wife, Zhu Ding, and our son, Roy. Beside supporting me in the everyday life, Zhu helped proof-read most of my manuscripts with many constructive comments. Roy came just when I began my doctoral study. For most of the time, he has been very positive and cooperative as well as active.

Contents

List of Figures

Chapter 1

Introduction and Problem Statement

Synchronization is an important step in establishing communication connections. Although it is often ignored by general users, the problem of synchronization must be considered by designers and appropriately handled in all types of communications. For example, a television must scan each line in accordance with the transmitter in order to rebuild each frame of video. There are 25 (PAL format) or 30 (NTSC format) frames every second. The television ought to synchronize each frame so that viewers can obtain fluent consistent images. Likewise, a cell phone user must synchronize with the base station in order to receive and make calls.

For a digital communication receiver, the procedure of synchronization consists of estimating the parameters of received signals besides the unknown data. Like other estimation problems, synchronization becomes more difficult when the signal is corrupted by noise. Nevertheless communications at very low signal-to-noise ratio (SNR), where signals are severely corrupted by noise, is of great interest because

1. It is desirable to keep transmission power at a level as low as possible in order to prolong a transmitter's battery life.

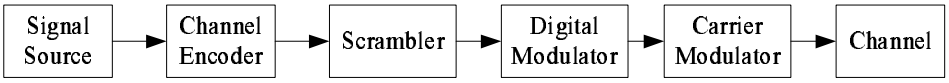


Figure 1.1: Diagram of digital transmitter.

2. Lower transmission power reduces inter-user interference.

3. For the overall system, lowering the required receive power helps to expand coverage and increase capacity.

Channel-capacity-approaching foward error-correction (FEC) codes, like turbo codes and low-density parity-check (LDPC) codes, are capable of achieving highly reliable communications at very low SNR. However, their performance is sensitive to synchronization errors. Their potential can only by obtained when proper synchronization is available. This dissertation focuses on symbol timing synchronization and frame synchronization for turbo codes and LDPC codes in very low SNR circumstances. The proposed techniques are applicable to communication systems with moderate data rate (below several Mega bits per second) and low to moderate code rate (below 1/2). New techniques are proposed to help coded systems recover the performance loss due to improper synchronization. The proposed solutions are suitable for applications in deep-space, satellite, fixed-wireless, or wireline communications.

1.1 System overview

Synchronization assures that the receiver is clocked in accordance with the transmitter. A diagram of a typical transmitter used within a digital communication system is shown in Fig. 1.1. Information is transmitted in the form of digital symbols. The source is a flow of random data. The symbols could be analog signals sampled and quantized to digital format, or inherently digital data. Forward error correction (FEC) codes are implemented to increase reliability by managing redundancy in the transmitted signals. A channel encoder adds

controlled redundancy into the source sequence so that errors can be corrected or detected at the receiver.

A scrambler, or channel interleaver, is used to break the correlation in the coded sequence. When the signal propagates through a fading channel, the received version tends to have burst errors. A scrambler helps to break the burst errors into random errors. It is needed when the coding method, like turbo coding, is not as good at correcting burst errors as correcting random errors. The scrambler also helps synchronization.

A digital modulator maps the scrambled sequence onto a complex constellation. Pulse shaping makes the transmitted signal comply with bandwidth constraints. Usually pulse shaping is implemented by concatenation of a transmit filter and a receive filter in the transmitter and the receiver respectively. Let $g_T(t)$ and $g_R(t)$ denote the impulse-response functions of the transmit filter and the receive filter. The pulse shape function $g(t)$ is a convolution of $g_T(t)$ and $g_R(t)$.

$$g(t) = g_T(t) * g_R(t) \tag{1.1}$$

If raised cosine (RC) roll-off pulse shaping is used, then the pulse shape function will be

$$g\left(t\right) = \operatorname{sinc}\left(\pi t/T\right) \frac{\cos\left(\pi\alpha t/T\right)}{1 - 4\alpha^2 t^2/T^2}. \tag{1.2}$$

where α is the roll-off factor and T is the symbol duration. When the pulse is longer than one symbol duration, then adjacent symbols may overlap, causing inter-symbol interference (ISI). One property of RC pulse shapes is that $g(0) = 1$, and $g(kT) = 0$, for $k \neq 0$, so it satisfies Nyquist's criterion for zero ISI [1]. A carrier modulator up-converts the base-band signal to an intermediate frequency (IF) and then a radio frequency (RF) suitable for propagation in wireless channels.

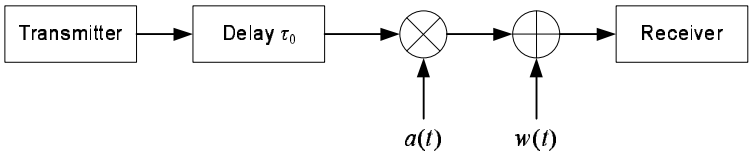


Figure 1.2: Channel model with time delay τ, flat fading gain $a(t)$, and additive noise $w(t)$.

The transmitted signal in baseband representation is

$$x(t) = \sqrt{E_s} \sum_{i=-\infty}^{\infty} d_i g_T(t - iT) \tag{1.3}$$

where $\{d_i\}$ is the coded symbol sequence. If binary phase-shift-keying (BPSK) modulation is used, then $d_i = \pm 1$. E_s is the transmitted energy per coded symbol.

The signal $x(t)$ is sent over a wireless channel, which affects the signal in several ways, as shown in Fig. 1.2. One characteristic of the channel is that it delays the signal by an unknown time τ, and it is the estimation of τ that is the main problem confronted by this dissertation. The time delay τ is decomposed into two parts

$$\tau = \mu T + \tau_s, \tag{1.4}$$

with μ an integer and τ_s in the range of $[-T/2, T/2]$. μ presents the shift of the symbols and is estimated by a frame synchronizer. τ_s denotes the timing offset within a symbol duration and is recovered by a symbol timing synchronizer.

In addition, the wireless channel acts as a linear time-varying filter. In the case of flat fading [2], the channel multiplies the signal by a time varying gain $a(t)$. Gaussian white noise $w(t)$ is then added to the received signal, and the noise spectral density is $N_0/2$. When $a(t) = 1$, the channel is only corrupted by additive white Gaussian noise (AWGN), and so such channels are simply called "AWGN".

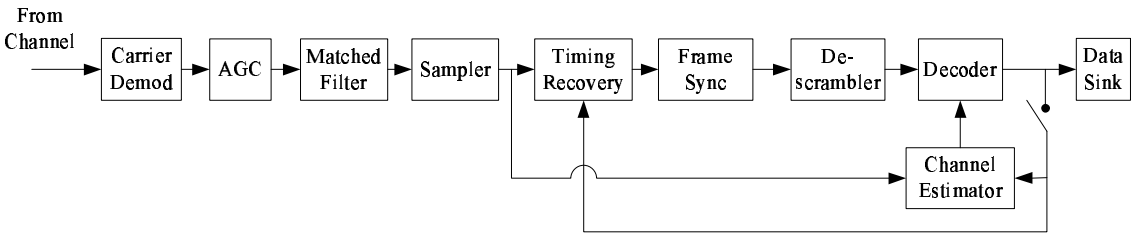


Figure 1.3: Diagram of digital receiver.

A receiver, shown in Fig. 1.3, picks up signals from the channel. The receiver is designed to recover transmitted data correctly. The receiver must be synchronized with the transmitter's carrier frequency, phase, symbol timing and frame timing. The carrier frequency and phase synchronization is usually achieved by inserting pilot tones and/or pilot symbols [3]. Achieving symbol and frame synchronization requires additional signal processing.

A carrier demodulator down-converts the RF signal to baseband for further processing. Perfect carrier frequency synchronization is assumed in this dissertation. An ideal automatic gain controller (AGC) normalizes the average magnitude of the received signal to a constant. In Fig. 1.3, the input of the matched filter is

$$y(t) = a(t)\, x(t+\tau) + w(t) \tag{1.5}$$

The matched filter has impulse response

$$g_R(t) = g_T(T-t). \tag{1.6}$$

The matched filter maximizes the output signal-to-noise ratio in an AWGN channel [4]. The output of the matched filter is

$$r(t) = \sqrt{E_s} \sum_{k=-\infty}^{\infty} a(t)\, d_k g(t - kT + \tau) + w_R(t). \tag{1.7}$$

The additive noise $w_R(t)$ becomes colored because of the matched filter and has autocorrelation function

$$R_w(\nu) = N_0 g(\nu). \tag{1.8}$$

$r(t)$ is sampled N times per symbol. N is normally an integer though we also consider non-integer values of N in Chapter 4. If the sampler fails to sample $r(t)$ at the correct timing, the receiver will suffer from signal energy loss and inter-symbol interference. A symbol timing synchronizer helps the sampler to estimate and track the timing offset τ within one symbol period. A frame synchronizer is inserted to find the starting point of a codeword. If the frame synchronizer fails to locate the correct position, the whole codeword is lost. The combination of the matched filter and the sampler is referred to as a digital demodulator. It is the counter part of the digital modulator in the transmitter.

While either a hard or soft decision could be made on the demodulator output, soft decisions are preferred because they provide more information to the decoder [1]. The descrambler permutes the received sequence back to its original order. The channel estimator estimates the channel state, including the propagation gain, phase (if the receiver is coherent), and noise variance. The channel estimator also provides SNR estimates which are needed for soft-input decoders that are required for typical decode capacity-approaching codes like turbo codes and LDPC codes.

The decoder seeks the most probable transmitted data sequence based on the demodulated data signals. Usually soft-input iterative maximum *a posteriori* (MAP) decoding processes are implemented to decode turbo codes and LDPC codes, using side information generated by the channel estimator. Decoding is iterative and greater coding gain is achieved after each round of iterations until the incremental increase in coding gain diminishes. The decoding process stops when it reaches a maximum number of iterations or the decoding

result has converged to a valid codeword. Intermediate decoding results are available after the first decoding cycle. The interim results enable the use of decision-feedback estimation for improved accuracy in symbol timing recovery, frame synchronization, and channel estimation.

A sink is the destination of data transmission, where the quality of performance is evaluated in terms of bit error rate (BER) and frame error rate (FER). Lower error rate correspond to more reliable connections.

Symbol timing synchronization and frame synchronization play important roles in the receiver. Existing techniques and issues in symbol timing synchronization and frame synchronization are introduced next.

1.2 Packet transmission and packet processing

One important setting used in this research is that coded data are transmitted in individual packets. This is a valid model used in time division multiplexing (TDM) systems, peer-to-peer communication systems, and *ad hoc* systems, where a global clock is unavailable. The receiver must synchronize with the transmitted packet based on the signals it receives.

In continuous transmission mode, all data are transmitted in a strict sequence. In contrast, a packet-based transmitter sends frames of data individually with pauses between transmissions, as illustrated in Fig. 1.4. The guard time is set long enough to prevent packets from overlapping. During the guard time, no data are transmitted, and the empty symbols during the guard times are referred to as "blanks". In this work, a packet contains one whole codeword that is either turbo coded or LDPC coded, and the terminology of "frame" is interchangeable with "packet" and "codeword".

The concept of packet processing using oversampling and interpolation is becoming attractive thanks to the availability of high-speed analog-to-digital converters and tremendous

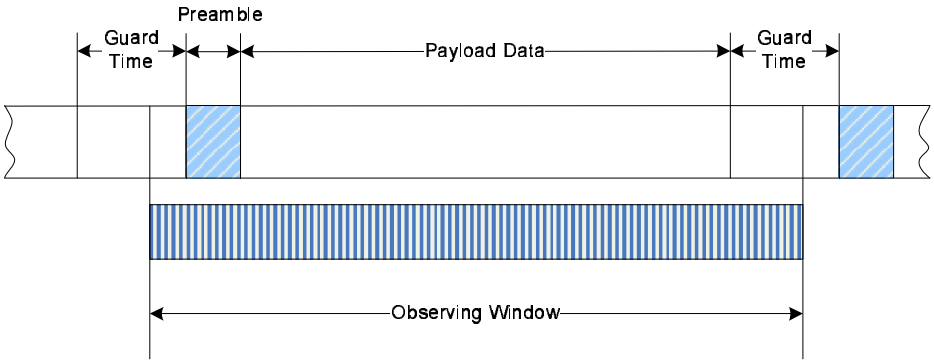


Figure 1.4: Packet transmission.

computational power. The received signal is sampled multiple times each symbol to preserve information including data and channel characteristics. Processing is restricted to a finite length observing window, which contains samples of the complete frame, as shown in Fig. 1.4.

In the remainder of this dissertation, we make the following assumptions:

1. The channel is AWGN. Therefore only additive Gaussian noise is considered.

2. The channel is quasi-static and so the SNR is fixed for a frame, but changes independently from frame to frame.

3. The time delay τ is unknown and may drift during a frame.

1.3 Symbol synchronization

The receiver must sample the output of the matched filter at the right timing. In a noise-free environment, an example matched filter output is shown for BPSK modulated data in Fig. 1.5. The transmitted data is [+1, -1, +1, -1, +1, +1, -1]. The pulse shaping uses raised cosine pulses with roll-off factor $\alpha = 0.5$. If the waveform is sampled with correct timing, as

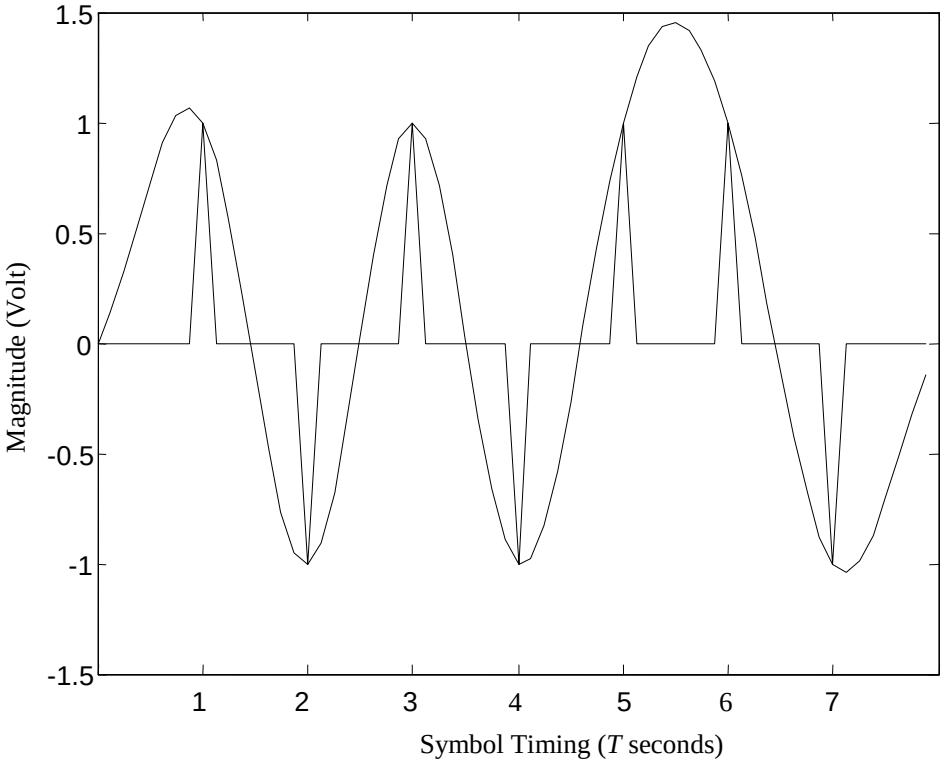


Figure 1.5: Waveform in noise-free channel. BPSK modulation is used. The transmitted data is [+1, -1, +1, -1, +1, +1, -1]. Raised cosine pulse shaping is used with roll-off factor $\alpha = 0.5$.

the pulses shown in Fig. 1.5, values with maximum SNR will be obtained. Otherwise the receiver will suffer from symbol energy loss and ISI.

1.3.1 Traditional symbol timing synchronization

Traditional real-time synchronizers consist of a sampler, a timing error detector (TED), a loop filter, and a voltage-controlled oscillator (VCO) as shown in Fig. 1.6. The TED generates $\hat{\tau}$, which is an estimate of τ, or, alternatively, the timing error $\epsilon = \tau - \hat{\tau}$. The looped filter is a low-pass filter over the observation window. The error signal is filtered by the loop filter and controls the VCO, which locks the sampler's rate and phase to the estimated timing information.

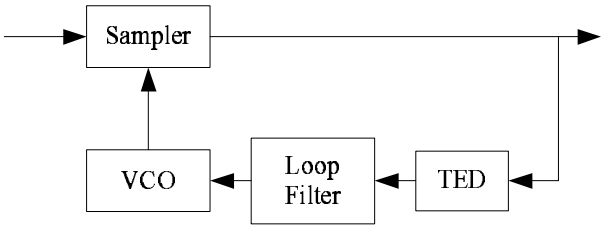


Figure 1.6: Symbol timing synchronizer in continuous time.

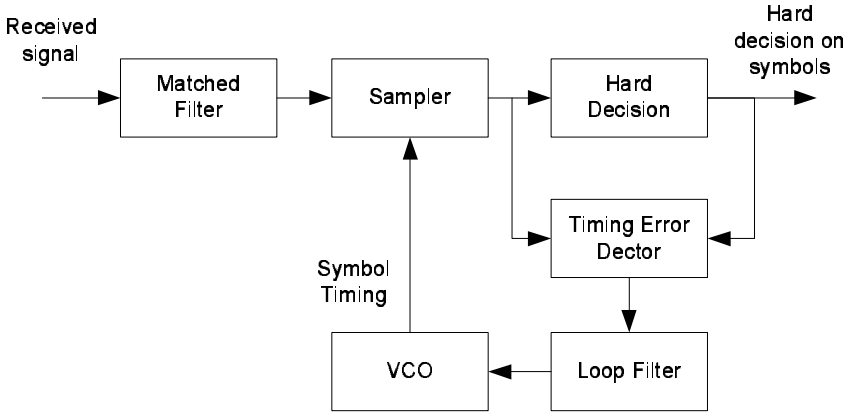


Figure 1.7: Mueller and Müller's synchronizer.

There are generally two main types of TED. One is decision-aided (DA) and the other is non-decision-aided (NDA). One widely accepted DA timing error detector is Mueller and Müller's method [5] as shown in Fig. 1.7. The matched filter is sampled at the symbol rate, *i.e.* only one sample per symbol is needed. A hard decision is made on the sample. Timing error is determined by comparing the value of samples and the hard decision results.

One category of NDA timing error detectors is derived from the technique proposed by Gardner [6]. This type uses more than one sample per symbol and is non-decision directed. A special case of NDA synchronizer is the early-late gate as shown in Fig. 1.8 [7]. In tracking mode, one of the samplers is tuned before the sampling time by δ, while the other is tuned after the sampling time by δ. The difference in the samples contains the sign and amplitude of the timing error.

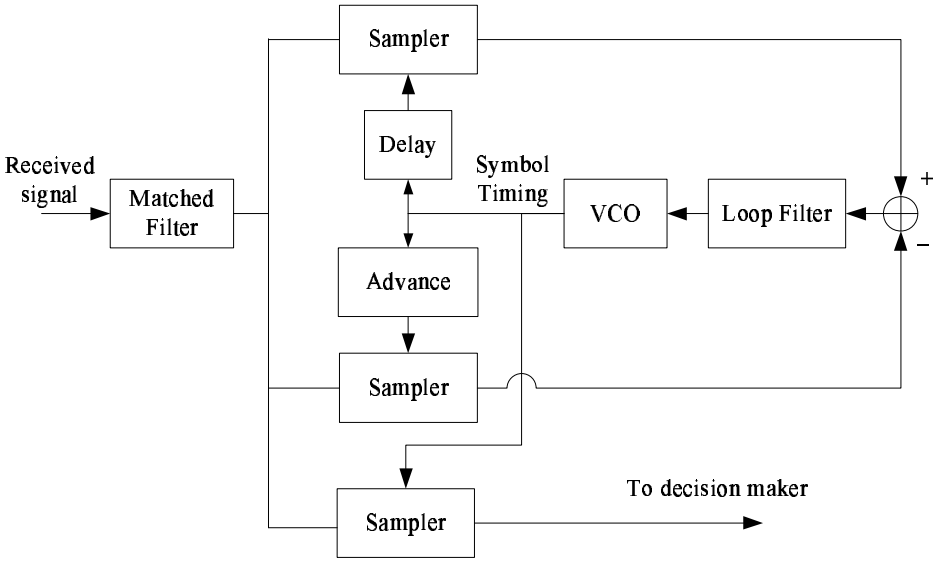


Figure 1.8: Early-late gate synchronizer.

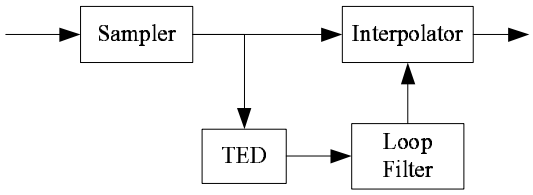


Figure 1.9: Symbol timing synchronizer using ITR.

1.3.2 Interpolation timing recovery

The Nyquist sampling theorem indicates that if the sampling rate is high enough (greater than the Nyquist frequency), then it is possible to reconstruct the original signal. Thereafter, the reconstructed signal can be resampled at the desired timing. The two steps of interpolation and resampling can be combined into one step to reconstruct the samples. This processing is called interpolation timing recovery (ITR). A diagram of an ITR-based synchronizer is shown in Fig. 1.9. A timing error detector (TED) estimates the time delay between the actual and ideal sampling time [8][9][10][11][12][13][14].

The interpolation method suggested by the sampling theorem is to use a sinc function

for reconstruction.

$$\text{sinc}\,(x) = \frac{\sin\,(x)}{x} \tag{1.9}$$

Polynomial functions are more practical than the sinc function. The simplest interpolation is zeroth order interpolation, which takes the sample value that is closest to the desired timing instance as the interpolation result. A better interpolation is linear interpolation, or first order interpolation. It returns a linear combination of the two closest samples to the desirable position. Although quite simple, it usually gives sufficiently accurate interpolation results. Higher order interpolations and other non-linear interpolation methods are more complicated and do not necessarily work better at low SNR. They are likely to boost up noise, and therefore are not suitable to be implemented for the applications of interest with very low SNR values.

The output of the matched filter $r(t)$ in (1.7) is sampled N times each symbol period. Assuming that the time delay is constant during the whole frame (later this assumption is relaxed), the sample sequence is

$$\begin{aligned} r\,[n] &= r\left(\frac{nT}{N}\right) \\ &= \sqrt{E_s}\sum_{k=-\infty}^{\infty} d_k g\left(\frac{nT}{N} - kT + \tau\right) + w_R\left(\frac{nT}{N}\right). \end{aligned} \tag{1.10}$$

Usually N is an integer between 2 and 4. Generally, the greater N is, the more accurate the timing estimate will be though this increases the system's complexity by requiring a higher rate analog-to-digital converter. Also the interpolation benefits from more information contained in more samples per symbol.

References [8] and [9] explore methods for estimating the time delay using sampling rates higher than twice the symbol rate. In [8], a square law device is used to remove the

uncertainty in the random data.

$$x[n] = |r[n]|^2 . \tag{1.11}$$

The frequency component at the symbol rate is found using a discrete Fourier transform (DFT)

$$X = \sum_{n=-NL_F}^{NL_F-1} x[n] e^{-j2\pi \frac{n}{N}} \tag{1.12}$$

where L_F is the size of an observation window. Due to a property of the DFT, the time delay in the time domain is transformed to the phase in the frequency domain [15]. Thus the time delay is estimated by observing the phase angle of the transformation result X,

$$\hat{\tau} = -\frac{T}{2\pi} \arg(X) . \tag{1.13}$$

The function $\arg(X)$ is the angle of X, with a value between $-\pi$ and π, and so $\hat{\tau}$ is in the range $[-\frac{T}{2}, \frac{T}{2})$. It is shown in [8] that the time delay estimate is unbiased. The operation resembles a digital phase-locked loop (PLL). The square law operation is a non-linear operation. Other operations like the absolute value and logarithm are also applicable [9][10].

Lee [10] proposed a method that requires only two samples per symbol and estimates the time delay as

$$\hat{\tau} = \frac{T}{2\pi} \left\{ \sum_{n=-2L_F}^{2L_F} \left[|r[n]|^2 e^{-jk\pi} + \Re\left[r(kT) r^*[(n-1)T] \right] e^{-j(n-0.5)\pi} \right] \right\} . \tag{1.14}$$

One issue with this TED is that the estimate is biased although the bias is very small. This issue is mitigated by a modification proposed by [12].

All the techniques described above assume that the time delay is changing slowly so that

it is constant during the observation interval. They also assume that the sampling rate is a known integer multiple of the symbol rate. However, the sampling time may drift randomly due to improper alignment and jitter in the local clocks at the transmitter and the receiver. Therefore the sampling time is not necessarily fixed. Rather, it wanders according to a random time walk. There are two types of random time walk. The first type, *linear walk*, is a result of using a fractional sampling rate. The second type, *random jitter*, characterizes the random drift of the sampling time. If time walk is not properly handled, it may severely degrade the receiver's performance.

1.4 Frame synchronization

A frame is a packet of data that conveys a certain amount of information. In this context, a frame contains a codeword. A frame synchronizer estimate μ in (1.4) and determines the starting point of a frame. Frame synchronization is required, in addition to other types of synchronization, to correctly receive the complete packet. The most straightforward method to achieve frame synchronization is to insert synchronization markers, known as sync words, into the transmitted sequence. The receiver keeps searching for the markers. If a marker is detected, then the receiver is ready to read the packet. Usually, the receiver uses a sliding window to calculate the correlation of the stored sync word pattern and the received signal. The candidate markers must have good autocorrelation properties, i.e., they have high peaks when the frame is synchronized and low side lobes when it is not. Barker codes and some other pseudo-random codes are usually used for their outstanding autocorrelation properties. The family of Barker codes is shown in Table 1.1.

Fig. 1.10 shows the autocorrelation property of a 13-bit Barker code. In a noiseless environment, the autocorrelation has a peak of 13 when synchronized. Otherwise, the correlation value is no greater than one. An alternative family of sync words which has longer length is

Length	Sync Word
2	1 0
3	1 1 0
4	1 1 0 1
5	1 1 1 0 1
7	1 1 1 0 0 1 0
11	1 1 1 0 0 0 1 0 0 1 0
13	1 1 1 1 1 0 0 1 1 0 1 0 1

Table 1.1: Barker Codes.

shown in Table 1.2.

A likelihood function is established to indicate the probability of frame synchronization at each position. Let μ be all possible starting points, and $L(\mu)$ be the likelihood function. The likelihood function for the correlation approach is

$$L(\mu) = \sum_{i=0}^{M_w-1} r[\mu + i] W_s[i] \tag{1.15}$$

where M_w is the length of the sync word and $W_s[i]$ represents the sync word. There are two commonly used decision rules, the *threshold rule* and the *maximum rule*. For the threshold rule, as illustrated in Fig. 1.11(a), the frame synchronizer compares the present $L(\mu)$ value against an optimized threshold. Once $L(\mu)$ is greater than the threshold, a frame sync is declared. By the maximum rule, as in Fig. 1.11(b) the index number corresponding to the maximum of all $L(\mu)$ is selected as the most reliable decision. Generally, the maximum rule is more robust than the threshold rule, but more complex.

One issue with using sync words is that a pattern that is identical or close to the sync word may also appear in the data. When this occurs, the frame synchronizer may give a wrong decision. Therefore the frame sync error rate is bounded by a lower limit, called the random data limit.

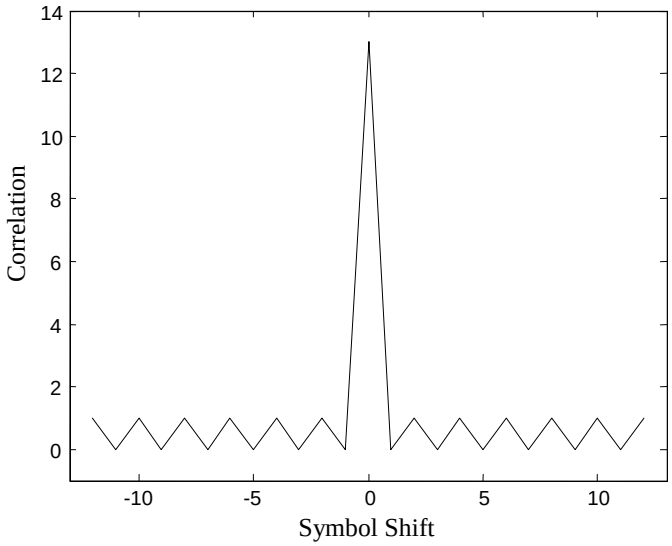
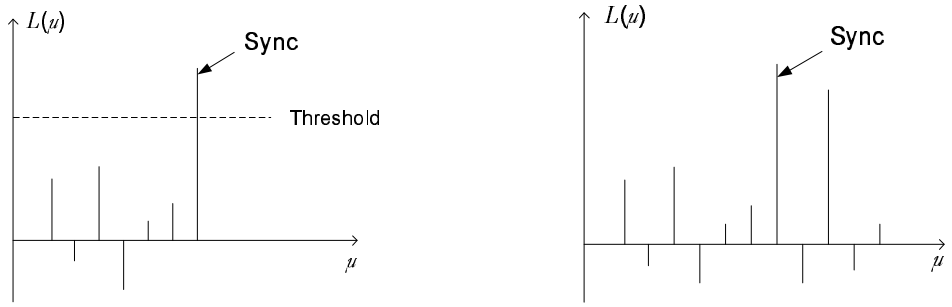


Figure 1.10: Auto-correlation property of 13-bit Barker code.



(a) Threshold rule.

(b) Maximum rule.

Figure 1.11: Decision of frame synchronizers.

In very low SNR circumstances, the signal is severely corrupted by noise. the correlation result calculated by the frame synchronizer also becomes noisy. The frame synchronizer may fail to detect the presence of a sync word.

1.4.1 Optimum frame synchronizer for continuous transmission

Optimum frame synchronization for continuous transmission is discussed by Massey for BPSK modulation in [17]. In continuous transmission, frames are transmitted sequentially

Length	Sync word	Length	Sync word
7	1011000	19	1111100110010100000
8	10111000	20	11101101111000100000
9	101110000	21	111011101001011000000
10	1101110000	22	1111001101101010000000
11	10110111000	23	10110101101011010000000
12	110101100000	24	11111010111001100100000
13	1110101100000	25	11111001011011100010000000
14	11100110100000	26	11111010011010011001000000
15	111011001010000	27	11111010110100110011000000
16	1110101110010000	28	11110101111001011001100000000
17	11110011010100000	29	111101011110011001101000000000
18	111100110101000000	30	1111101011110011001101000000000

Table 1.2: Alternative sync words [16].

and the sync words are surrounded by random data as shown in Fig. 1.12. A maximum likelihood (ML) synchronizer is proposed as a modification to the correlation approach in (1.15), taking into account the data that are adjacent to the sync words. Two approximate algorithms for high SNR and low SNR are provided to reduce the complexity of the receiver.

Lui and Tan [18] extended the ML decision rule to M-ary modulated constellation. Several bounds are also given in [18], particularly the random data limit.

1.4.2 Frame synchronization for coded systems

The frame synchronization methods presented above only consider random data, i.e., the code structure in the received sequence is ignored. If the data are coded, it should be realized

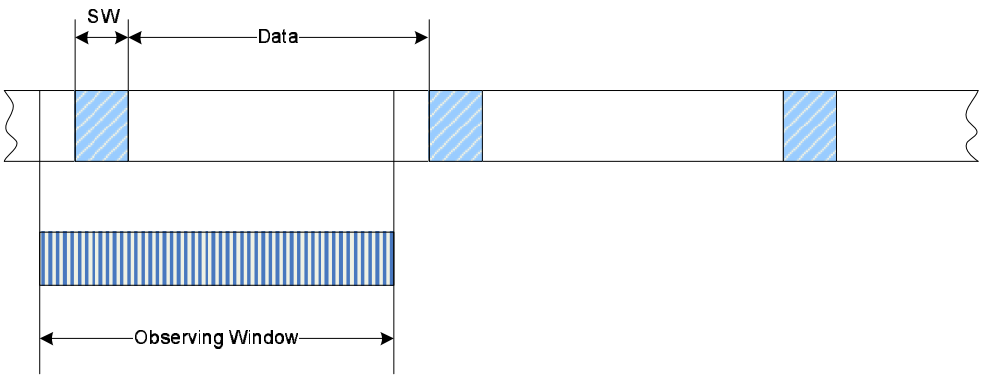


Figure 1.12: Frame synchronization in continuous transmission mode.

that the frame synchronizer and the decoder can be integrated into a joint estimator so that sync words are not absolutely necessary. Intuitively frame synchronization is achieved if, and only if, the received frame corresponds to a valid codeword. Robertson [19] developed a generalized frame synchronizer, taking into account general coded data. The method improves the performance by several dB at high SNR.

1.4.3 Packet-transmission and preamble-less frame synchronization

In many systems, data are transmitted in separate packets instead of continuous streams. Robertson [20] discussed optimum frame synchronization for preamble-less packet transmission. The synchronizer has similar structure as those for continuous transmission. However, the blanks before and after each packet need to be considered.

Howlader and Woerner [21] [22] applied list synchronization to packet-transmission of convolutional codes and turbo codes. The code structure, trellis in this case, is considered. Sync words are coded to be included into the trellis so that the decoder works in synergy with the synchronizer to detect the correct position of the codeword.

Cassaro and Georghiades [23] proposed the most general frame synchronizer so far, which

combines a low complexity synchronizer running a threshold rule and a high complexity part implementing list synchronizer. The former treats the incoming data as uncoded symbols with the algorithm provided in [18]. If the synchronizer at this stage gives a confident estimate about frame synchronization, then the high complexity part is by-passed. Otherwise, several candidates with the most significant likelihood are recorded and a decoder is used to examine the candidates and select the most likely frame sync position. Recursive systematic convolutional (RSC) codes are considered in [23]. RSC codes are inherently cyclic codes, therefore a channel interleaver is used to disturb the regular distribution of codewords.

Matsumoto and Imai [24] proposed a frame synchronizer that explores iterative decoding for packet transmission of LDPC codes. The Gaussian approximation of sum-product decoding [25] is used on the variable nodes after one full iteration. Then the decoder determines if a potential codeword has been processed. If not, the data are shifted by one symbol and the decoder repeats the decoding-estimation cycle until a possible codeword is detected.

1.5 Proposed method and dissertation organization

The categories of synchronization and existing techniques are briefly introduced above. Issues are encountered when deploying powerful error-correction codes, like turbo codes and LDPC codes, in very low SNR environments. The goal of this dissertation is to find methods of estimating τ at the low SNR encountered by capacity-approaching codes. A key theme is that the structure of the code itself is exploited whenever possible.

After giving an overview of capacity-approaching codes in Chapter 2, the dissertation considers the following issues.

1.5.1 Joint estimation of time delay and SNR

As shown in Fig. 1.3, MAP decoding of turbo codes and LDPC codes requires knowledge of the channel. The samples $r[n]$ provide a sufficient statistic for computing the time delay and channel information. Therefore it is desirable to jointly estimate the time delay and the channel state. Decision-feedback estimation is implemented to achieve better performance. Such a joint approach for turbo codes in AWGN channels is proposed and analyzed in Chapter 3. The results were published in IEEE International Conference on Communications 2003 [14], and is accepted for publishing in IEEE Transactions on Communications [26].

The proposed technique is applicable to flat Rayleigh fading channels where the coherence time is longer than the duration of one frame. The channel is quasi-static which remains fixed during the interval of one codeword and changes from frame to frame.

1.5.2 Mitigation of time walk

Random time walk has negative impact on symbol timing estimation. Time walk over a long observation window may accumulate to significant timing offset. In extreme cases, this may cause cycle slips, and consequently, irreducible burst errors. Its effect is mitigated by overlapped sliding windows and iterative timing refinery as proposed in Chapter 4 and is submitted to IEEE Vehicular Technology Conference Spring 2005 [27].

1.5.3 Preamble-less frame synchronization

If sync words are used, they do not convey any information. Due to this fact, the power efficiency of the system is lowered. This energy loss from sync word insertion is critical whenever capacity-approaching error correction coding is used in very low SNR. Therefore, it is desirable to find a frame synchronization method that does not require insertion of sync words.

A frame synchronizer for packet-transmission of LDPC codes is developed where no sync word or preamble is required as described in Chapter 5. The proposed method is to exploit the inherent parity-check properties of LDPC codes and turbo codes to detect the starting point of a frame. The algorithm is based on Gaussian approximation of check nodes and only requires half of a decoding iteration. The complexity of the proposed method is approximately half that of [24]. It is able to replace the high-complexity component in [23] in cases that data are LDPC coded or turbo coded. The results are accepted to be published in Asilomar Conference on Signals, Systems, and Computers 2004 [28].

1.5.4 Integrated system with timing and frame synchronization

We conclude the dissertation by considering an integrated receiver for a turbo coded system in Chapter 6. The receiver consists of a symbol timing synchronizer, a SNR estimator, a frame synchronization, and a turbo decoder. The results verify the proposed techniques as promising approaches to implement efficient receivers for capacity-coded systems.

Chapter 2

Capacity-Approaching Coding

This chapter reviews fundamentals of channel capacity and error correction coding. The code structure and iterative decoding of turbo codes and LDPC codes are described.

2.1 Channel capacity

The channel capacity C, introduced by the milestone paper of Shannon[29], is the upper limit of the data rate that can be reached on a noisy channel. If an arbitrary signal is used, the highest data rate on a Gaussian channel is

$$\mathsf{C} = \frac{1}{2}\log_2\left(1 + \frac{2E_s}{N_0}\right) \tag{2.1}$$

The capacity theory states

1. If the data rate $R > \mathsf{C}$, then there is no possibility to realize error-free data transmission;

2. If $R \leq \mathsf{C}$, then there exists a coding scheme to achieve reliable transmission with arbitrarily small error probability.

The capacity theory itself does not give a solution to how to achieve the capacity. The proof of the theory suggests that a random code should be used to achieve the capacity. However, a random code is not realizable because it lacks structure. For quite a long time, the best performance any code could reach was still a few dB away from the theoretic capacity. It was in the 1990's when iterative decoding made the historic leap in coding theory. Turbo codes and low density parity check (LDPC) codes were demonstrated to be capable of reaching less than 1 dB from Shannon's capacity [30] [31]. Turbo codes and LDPC codes represent practical strategies for approaching capacity by exploiting pseudo-random code structure and iterative decoding. Turbo codes are constructed on concatenated convolutional codes. LDPC codes are an ensemble of linear block codes that have sparse parity check matrices. Fundamentals of coding theory are briefly reviewed below.

2.2 Error-correction codes

Error correction coding is the process of adding redundancy into the data in order to protect the information from channel errors. A decoder is required at the receiver, which exploits the structure of the redundancy to estimate the correct data. There are two basic types of error correction codes, namely, linear-block codes and convolutional codes.

2.2.1 Linear block codes

The linear block codes are based on linear algebra on Galois fields. A linear block code is specified by the generating matrix $\mathbf{G}$ (or the parity check matrix $\mathbf{H}$). The data is represented by a row vector $\mathbf{x}$ of n_k bits. The operation

$$\mathbf{c} = \mathbf{xG} \tag{2.2}$$

expands the space of data bits into a codeword space with a dimension of n_n, where $\mathbf{G}$ is a $n_k \times n_n$ matrix. If a binary code is considered, then among all 2^{n_n} codes, only a set of 2^{n_k} codewords are valid. The distance between valid codewords is defined as the *Hamming distance*. And the smallest Hamming distance is the *minimum distance*, denoted by d_{min}. d_{min} determines the capability of a linear block code to correct and detect errors. The larger d_{min} a code has, the more powerful the code is.

The conventional decoding of a linear block code is based on linear algebra in finite Galois fields [32]. $\mathbf{H}$ consists of element vectors in the null-space of $\mathbf{G}$, which means

$$\mathbf{G}\mathbf{H}^T = 0. \tag{2.3}$$

The dimension of $\mathbf{H}$ is $n_m \times n_n$, where $n_m = n_n - n_k$. $\mathbf{H}$ defines a set of parity check equations. If $\mathbf{c}$ is a valid codeword, then it must satisfy

$$\mathbf{c}\mathbf{H}^T = \mathbf{0} \tag{2.4}$$

Hard decisions on the received signals are made prior to decoding. The hard decision result $\hat{\mathbf{c}}$ is the transmitted codeword $\mathbf{c}$ disturbed by an error vector $\mathbf{e}$.

$$\hat{\mathbf{c}} = \mathbf{c} + \mathbf{e} \tag{2.5}$$

Because $\mathbf{c}\mathbf{H}^T = \mathbf{0}$, we obtain

$$\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T. \tag{2.6}$$

The decoder finds out $\mathbf{e}$ according to $\mathbf{e}\mathbf{H}^T$ and flips the corresponding bits to correct errors. Decoding errors happen when there are more than $d_{min}/2$ errors in $\hat{\mathbf{c}}$.

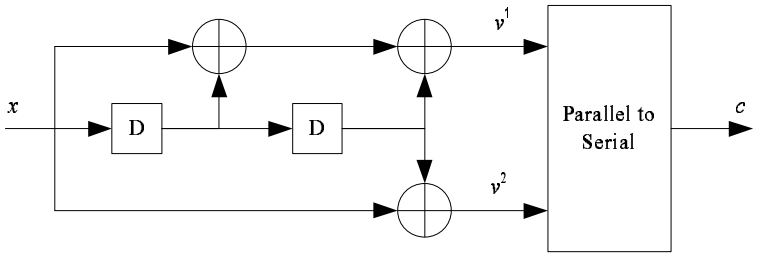


Figure 2.1: The encoder of (7, 5) NSC code.

2.2.2 Convolutional codes

A convolutional code is defined by a finite state machine. Usually the encoder is implemented with shift registers. Parameters of a convolutional code include the constraint length k_c and code rate r. k_c is the number of symbols taken into the convolutional operation. It is the number of registers plus input. Fig. 2.1 shows the encoder of a non-systematic convolutional (NSC) code. This code has two registers and one input, therefore it has $k_c = 3$. The encoder takes in one bit and outputs two bits each symbol period, thus it has a rate $r = 1/2$.

In Fig. 2.1, $\mathbf{v}^1$ and $\mathbf{v}^2$ are results of two bitwise convolutional operations. The convolutional coefficients are $(1, 1, 1)_2$ and $(1, 0, 1)_2$ respectively. The subscript $()_2$ indicates binary expressions. The code is represented in octal format as $(7, 5)_8$. A parallel-to-serial (P/S) convertor assembles $\mathbf{v}^1$ and $\mathbf{v}^2$ into the codeword sequence $\mathbf{c}$. In practice, the codeword is truncated to a given length.

The Viterbi algorithm (VA) [33] is the most widely used method for performing maximal likelihood sequence estimation (MLSE). An alternative decoding technique is introduced by Bahl *et al* in 1974 [34]. The so-called BCJR algorithm performs maximum *a posteriori* (MAP) detection of each data symbol. Although the complexity of the BCJR algorithm is higher than VA, the BCJR algorithm found its application in turbo codes [30]. The BCJR algorithm is more suitable for turbo decoding because it can return soft output information about the probability of each data symbol, while the standard VA only returns decisions on

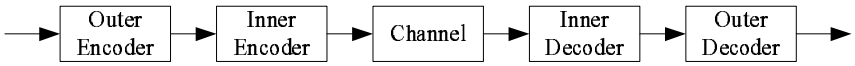


Figure 2.2: A diagram of concatenated codes.

the most likely sequence.

2.2.3 Concatenated codes

Concatenated codes provide a way to improve the error-correction capacity using realizable complexity [35]. A diagram of concatenated codes is shown in Fig. 2.2. Usually a concatenated code uses a convolutional code as the inner code and a Reed-Solomon code as the outer code[32] because the convolutional code is good at correct random errors and the Reed-Solomon code is capable of correcting burst errors.

The cascaded code in Fig. 2.2 does not have feedback from the outer decoder to the inner decoder. The two decoders work separately. The turbo codes modifies the code structure so that the two decoders work in synergy to improve the error-correction power in an iterative way.

2.3 Turbo codes

Turbo codes are a family of parallel concatenated convolutional codes (PCCC). They are featured constituent recursive systematic convolutional (RSC) codes, random interleavers, and iterative decoding.

2.3.1 Encoder

A typical structure of turbo codes consists of two identical constituent recursive systematic convolutional (RSC) codes. A rate 1/2 RSC code is shown in Fig. 2.3. It has one systematic

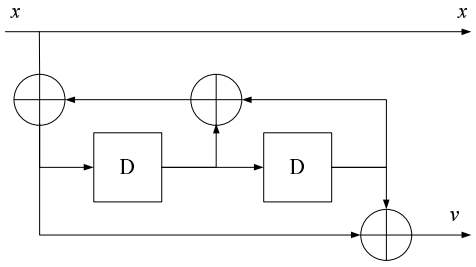


Figure 2.3: A (7, 5) RSC code.

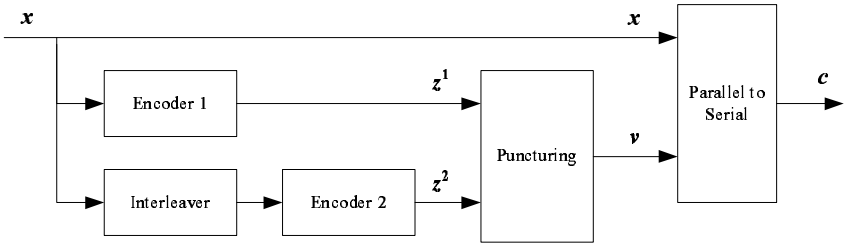


Figure 2.4: Encoder schematic of turbo codes.

output and one parity output. There are two branches of convolution operations. One of the two branches is fed back and exclusive-ored with the systematic input. This feedback mechanism makes the encoding process recursive.

As shown in Fig. 2.4, the information bits are fed into Encoder 1 to produce parity bits $\mathbf{z}^1$. An interleaver reorders the information bit sequence randomly or pseudo-randomly. The interleaved sequence is taken into Encoder 2 to generate $\mathbf{z}^2$. Since the systematic part of Encoder 1 and 2 contains the same information, only one copy of $\mathbf{x}$ is transmitted. The code rate is then approximately 1/3. The puncturing block periodically deletes certain parity bits to tailor the code rate to a proper higher rate. A P/S block outputs the codeword $\mathbf{c}$.

2.3.2 Decoder

Assuming an AWGN channel, the received codeword $\mathbf{y}$ is a sequence of transmitted data signal distorted by additive noise. If BPSK modulation is used, then

$$\mathbf{y} = 1 - 2\mathbf{c} + \mathbf{w} \tag{2.7}$$

where $\mathbf{w}$ is a vector of independent identically distributed (i.i.d.) additive noises with variance $\sigma^2 = N_0/2$. A serial-to-parallel convertor converts $\mathbf{y}$ into $\hat{\mathbf{x}}$ and $\hat{\mathbf{v}}$, where $\hat{\mathbf{x}}$ is a soft decision about data bits $\mathbf{x}$ and $\hat{\mathbf{v}}$ is a soft decision about parity bits $\mathbf{v}$. A bit-insertion block inserts dummy symbols in the places of punctured parity bits. The receiver has two soft-input soft-output (SISO) RSC decoders as shown in Fig. 2.5. The soft input refers to $\hat{\mathbf{x}}$ and extrinsic information $\mathbf{L}_e^i$ from another decoder about information bits. The soft output is usually the log-likelihood ratio (LLR) $\mathbf{\Lambda}$ about data bits, given by

$$\Lambda_i = \log\left(\frac{\Pr\left(x_i = 1 \,|\mathbf{Y} = \mathbf{y}\right)}{\Pr\left(x_i = 0 \,|\mathbf{Y} = \mathbf{y}\right)}\right). \tag{2.8}$$

The hard decision on Λ_i is

$$\bar{x}_k = \begin{cases} 1 & \text{if } \Lambda_i > 0 \\ 0 & \text{if } \Lambda_i < 0 \end{cases} \tag{2.9}$$

The LLR derived from observations of the channel is $2\mathbf{y}/\sigma^2$ in AWGN channels, where σ^2 is the variance of the additive noise. The extrinsic information output from the decoder is defined as

$$\mathbf{L}_e^o = \mathbf{\Lambda} - 2\hat{\mathbf{x}}\,/\sigma^2 - \mathbf{L}_e^i\,, \tag{2.10}$$

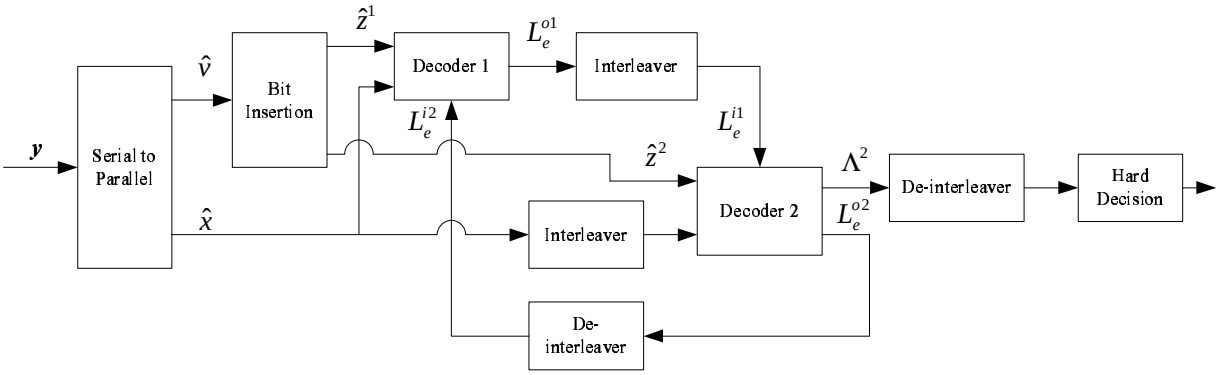


Figure 2.5: Decoder schematic of turbo codes.

which is the pure decoding gain achieved from each individual RSC decoder.

The BCJR algorithm is described in detail in Appendix C [36]. For the turbo decoding process, in the initial state, there is no extrinsic information. Decoder 1 starts decoding on $\hat{\mathbf{x}}$ and $\hat{\mathbf{z}}^1$, and outputs $\mathbf{L}_e^{o1}$. Decoder 2 takes $\mathbf{z}^2$ and properly interleaved $\hat{\mathbf{x}}$ and $\mathbf{L}_e^{o1}$, and generates soft decisions $\mathbf{\Lambda}^2$ and $\mathbf{L}_e^{o2}$. This finishes one iteration. $\mathbf{L}_e^{o2}$ is then passed to Decoder 1 and so starts a new iteration. A given number of iterations are processed before a hard decision is made on $\mathbf{\Lambda}^2$. The accumulation of extrinsic information helps the decoding result converge to the correct codeword.

2.4 LDPC codes

LDPC codes are an ensemble of linear block codes that have low density parity check matrices. They are more clearly defined by Tanner graphs. LDPC codes were first presented in Gallager's pioneer work in 1961 [37]. They were generally ignored by the coding society until in the late 1990's they were re-discovered by MacKay [31] and Urbanke [38]. LDPC codes are capable of correcting all errors if the Tanner graph is cycle-free.

2.4.1 Code structure

An LDPC code is a linear block code described by its parity check matrix $\mathbf{H}$. $\mathbf{H}$ is sparse, which means that most entries in $\mathbf{H}$ are "0", and it has very few "1" in each row and column. The dimension of $\mathbf{H}$ is $n_m \times n_n$. Let W_c be the number of "1" per column, and W_r the number of "1" per row, where $W_r << n_n$.

The generating matrix $\mathbf{G}$ consists of element vectors in the null-space of $\mathbf{H}$. One straightforward way to find $\mathbf{G}$ is to use row operations (and possible column permutations) to change $\mathbf{H}$ into a systematic form $\mathbf{H} = \left[\mathbf{P}^T \vdots \mathbf{I} \right]$, then $\mathbf{G} = \left[\mathbf{I} \vdots \mathbf{P} \right]$. $\mathbf{G}$ obtained from this process is generally no longer low-density. Hence encoding a LDPC code may have high complexity [39]. For example, considering a (10000, 5000) LDPC code, the size of $\mathbf{P}$ is 5000×5000. We may assume that the density of 1's in $\mathbf{P}$ is 0.5, then there are 12.5×10^6 1's in $\mathbf{P}$, so 12.5×10^6 exclusive-or operations are required to encode one codeword.

Conventional syndrome-based decoding methods are not suitable for LDPC codes because

1. $\mathbf{H}$ has a very large size.

2. Randomly generated $\mathbf{H}$ is not subject to structural decoding.

3. Hard-decision decoding is inferior to soft-input decoding because channel information is partly lost which results in an SNR loss of about 2 dB.

A soft-input *message-passing* algorithm based on the Tanner's graph [40] is applicable and contributes to the outstanding error-correcting power of LDPC codes. The following sections introduce how to describe LDPC codes with graphs and the message-passing decoding algorithm.

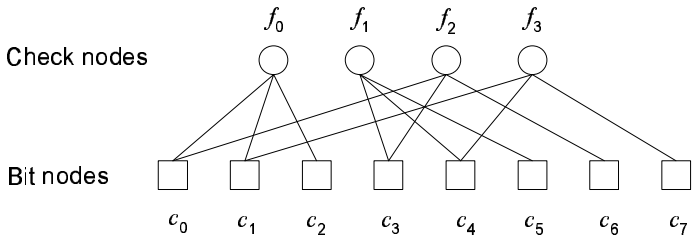


Figure 2.6: Tanner graph of a product (4, 8) codes.

2.4.2 Graph-based codes

The message-passing algorithm is based on a bipartite graph [38]. The bipartite graph, frequently referred to as Tanner graph [31], is a graphical interpretation of the parity check equations. It is bipartite because the vertices are divided into two groups. Every vertex, or node, in one group is only connected to nodes in the other group, and none of the same group. In a Tanner graph, the *check nodes* represent the parity check equations and the *variable nodes* denote the symbols in a codeword. The check node i is connected to variable node j iff $h_{ij} = 1$ in $\mathbf{H}$.

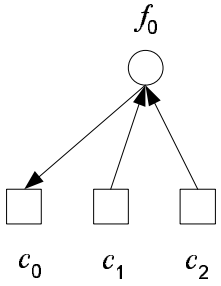
For example, consider an (8, 4) product code

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} \tag{2.11}$$

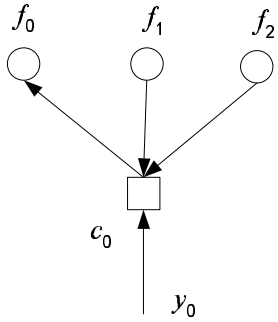
The corresponding Tanner graph of (2.11) is shown in Fig. 2.6.

2.4.3 Message-passing algorithm

The message passing algorithm is illustrated in Fig. 2.7 for the (8, 4) product code with the parity check matrix in (2.11) and Fig. 2.6. The algorithm finds MAP decision of each



(a) Message-passing to a variable node.



(b) Message-passing to a check node.

Figure 2.7: The message-passing algorithm.

symbol in a codeword based on the received signal. The information held by a check node is the likelihood that the parity check equation is satisfied, that is, an even number of "1"s are connected to the check node. For a variable node, the information is the likelihood that "1" or "0" is transmitted. The message passed around in the decoding process is the extrinsic information about each variable node (check node) from information gathered from check nodes (variable nodes) connected to it. As shown in Fig. 2.7(a), the message on path $f_i \rightarrow c_j$ is extrinsic information collected from other variable nodes connected to f_i. As shown in Fig. 2.7(b), the message on path $c_j \rightarrow f_i$ is formed by collecting extrinsic information from other check nodes connected to c_j and information from the channel.

Each iteration of the message passing decoding algorithm has two steps. The information on each variable node is initialized as the soft decision for the corresponding symbol from the channel. If an AWGN channel is considered, then it is $2y_0/\sigma^2$. In the first step, as shown in Fig. 2.7(a), every check node collects extrinsic information from variable nodes connected to it, and update the message on path $f_i \rightarrow c_j$. The second step is to update the message on paths from variable nodes to check nodes as shown in Fig. 2.7(b).

The likelihood ratio of each variable node is updated after each full iteration. Hard decisions $\hat{\mathbf{c}}$ are made on the codeword symbols. If the parity check equation $\hat{\mathbf{c}}\mathbf{H} = \mathbf{0}$ is satisfied, or the limit of iterations has been reached, then the decision is output to the

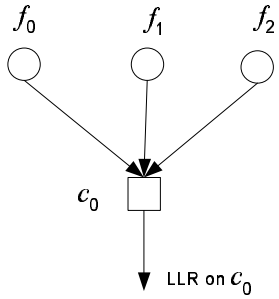


Figure 2.8: Update likelihood on variable nodes.

decision maker. Otherwise, the decoder starts another iteration. The details of the decoding algorithm in the probability domain are described in Appendix D.

2.5 Conclusion

One important similarity in turbo codes and LDPC codes is that the decoding process is iterative, fulfilling the turbo principle. The iterative processing feeds back decoding results to facilitate another round of iteration. Greater coding gain is achieved in each round until there is no more significant increment. The decoding stops when the decoding result converges to a valid code word, or the decoder has reached maximum iteration.

This inherent feedback characteristic in turbo codes and LDPC codes enables a library of decision-feedback estimations. The symbol timing estimation and channel estimation are among them. The intermediate decoding results are used in these estimators. Once the system converges, the feedback mechanism improves the accuracy of the estimators, the reliability of the decoder and, ultimately, improves the system performance.

Chapter 3

Joint Symbol Synchronization and SNR Estimation

This chapter proposes a method to jointly estimate symbol timing delay and SNR using samples from a codeword. The sampling rate is two to four times the symbol rate. The samples form a sufficient statistic of both the time delay and SNR. Estimates of time delay and SNR are found by a feedforward minimum mean square error algorithm. When a time delay estimate is available, proper timing is recovered by interpolation. SNR estimates provide channel reliability information to the soft-input decoder. Applying the turbo principle to the joint estimation algorithm, accuracy of timing and SNR is refined by decision feedback estimation.

The chapter begins with a brief review of existing techniques and lower bounds on estimation errors. The proposed system structure is described. The chapter then proceed to introduction of effective SNR, which is an important concept in developing the joint estimation algorithm. The joint estimation algorithm is analyzed and verified by simulation. It is shown that the proposed method can recover most of the coding gain loss of turbo coded systems due to random time delay.

3.1 Existing techniques

Although symbol timing synchronization is critical to the performance of turbo coded and LDPC coded systems, the problem is generally overlooked by researchers. Most publications assumes that perfect symbol timing is available. Nevertheless, a handful researchers have published their relative work.

3.1.1 M & M method

Liu *et.al.* [41] and Nayak *et.al* [42] discuss the performance of Mueller and Müller's synchronizer with a phase-locked-loop (PLL) and use it to track timing in systems that are protected by turbo or low density parity check (LDPC) codes. The authors consider specific applications in magnetic storage, where the codeword length is no longer than 1000 symbols and E_b/N_0 is around or above 5 dB. Iterative processing can help to refine timing estimation using intermediate decoding results from turbo decoders. Nayak *et.al.* [43] derived a lower bound for iterative timing recovery with a PLL-based structure, which has a gap of 7 dB to the CRB.

3.1.2 ITR method

Wu *et. al.* [44] discuss the effect of ITR using the minimum mean square error (MMSE) algorithm over a sliding window. The objective of ITR is to reconstruct the signal given the samples and knowledge of the timing offset associated with these samples. The accuracy of the estimator increases with larger window sizes and higher SNR, but the complexity of this algorithm grows exponentially with the window size. Hence it is not suitable for low SNR, where a large window size is required.

3.1.3 Early-late gate method

Lu and Wilson [45] proposed a front-end synchronizer that uses a combination of an early-late-gate and decision-feedback. Hard decisions on the code symbols are fed back to the synchronizer prior to decoding, and therefore the turbo decoder structure is not exploited. The authors assume perfect initial timing, and use a PLL to track changes in the timing.

3.1.4 Soft information combining

Mielczarek and Svensson [13] investigated the distribution of extrinsic information within a turbo decoder as a function of timing offset and introduced a soft-bit combining method for synchronization that employs two separate turbo decoders and generates the likelihood of each data bit by appropriately combining the two decoder's soft outputs. While this scheme provided performance within 0.2 dB of that with perfect timing, it did so at the cost of two decoders and sampling at four times the symbol rate and it assumed perfect SNR estimates. This complexity might not be affordable for many receivers. Furthermore, [13] only assumed a frame size of 256 information bits and a code rate of 1/2, which translated to a very weak turbo code that operated at high SNR where timing synchronization is less challenging.

3.1.5 Desirable method

The desirable symbol timing synchronization techniques are applicable for low-rate, capacity-approaching turbo codes operating over AWGN channels with random timing offset, band-limited pulse shapes, and modest data rates. The solutions are suitable for deep-space, satellite, fixed-wireless, or wireline communications. The goal of this study is to develop a synchronization algorithm with performance that is comparable to the one proposed in [13] but requires only a single turbo decoder.

3.2 Cramer-Rao bound

Because of noise and other impairments, the time delay estimate $\hat{\tau}$ is a random variable. A good estimator should give an unbiased estimate of the timing offset, i.e., $\mathbf{E}[\hat{\tau}] = \tau$. A well-designed estimator should also have a small variance $\sigma_{\hat{\tau}}^2 = \mathbf{E}\left[(\hat{\tau} - \tau)^2\right]$. The Cramer-Rao bound (CRB) defines a lower bound on the variance of the estimation error an estimator can achieve, regardless of any other unknown parameters [46]. In other words, it sets the limit for the precision of estimation, and also provides a benchmark for estimating the timing offset when the SNR is known.

The CRB for timing estimation is addressed by Georghiades and Moeneclaey [47], and Mengali and D'Andrea [11]. The normalized CRB is

$$\frac{\sigma_{\hat{\tau}}^2}{T^2} \geq \frac{1}{8\pi^2 \xi L} \cdot \frac{1}{E_s/N_0}, \tag{3.1}$$

where L is the observing window size, i.e. the number of symbols used in the estimator. ξ is a constant related to pulse shaping. For RC pulse shaping

$$\xi = \frac{1}{12} + \alpha^2 \left(\frac{1}{4} - \frac{2}{\pi^2} \right). \tag{3.2}$$

When the roll-off factor $\alpha = 0.5$, the normalized CRB of timing estimation is

$$\frac{\sigma_{\hat{\tau}}^2}{T^2} \geq \frac{N_0}{E_s \left(\frac{7}{6}\pi^2 - 4 \right) L}. \tag{3.3}$$

As shown in Fig. 3.1, the CRB is inversely proportional to the SNR and the window size. This implies that the variance $\sigma_{\hat{\tau}}^2$ can be decreased by increasing the SNR or using a larger observation window. When $E_s/N_0 = 0$ dB, the normalized CRB of symbol-by-symbol ($L = 1$) timing estimate is 0.1331. However, the lower bound is 1000 times smaller when we

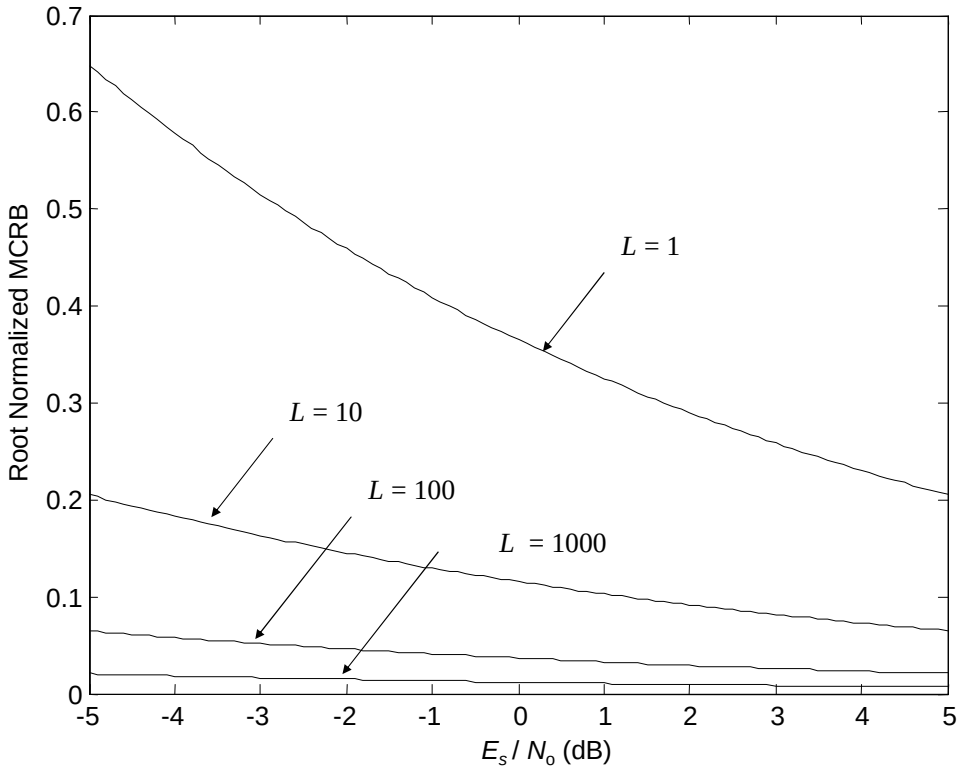


Figure 3.1: Root normalized CRB for different window sizes and raised-cosine roll-off factor $\alpha = 0.5$.

base our estimate on a data sequence whose length is 1000 symbols.

3.3 Joint time delay and SNR estimation

The turbo coded system considered in this work is shown in Fig. 3.2. The system consists of a matched filter. The output of the matched filter is sampled N times per symbol period. N is assumed to be an integer with typical value between two and four. For clarity, the diagram shows N samplers, each clocked at the symbol rate at sample instants that are shifted from one sampler to the next by T/N. A practical *synchronous sampling* implementation would use a single sampler clocked at N times the symbol rate [6]. Each of the N samplers passes its samples of the entire codeword to a block that calculates an online statistic for that sampler. This online statistic is related to effective SNR. All the samples are stored in memory according to their sampling order, forming N sample sequences. The online statistics are then passed to an estimator that jointly performs timing estimation and SNR estimation. With the switches being placed in position 1, the timing estimate is used to control an interpolator which combines the samples from the matched filter to yield a sufficient statistic for each symbol.

The decoder uses the interpolated samples as input. The turbo decoder is implemented with the *maximum a posteriori* (MAP) algorithm [30], so it needs an SNR estimate. We assume that the receiver has perfect carrier and phase synchronization. Furthermore, we assume that perfect frame synchronization is achieved so that the channel is quasi-static in the sense that it behaves as an AWGN channel for the duration of the frame and that the timing offset is constant for the entire frame (although the channel SNR and timing offset may vary from frame to frame).

The part enclosed by dotted lines is the subsystem for decision-aided estimation. Due to the iterative nature of turbo processing, timing and SNR estimates can be refined using

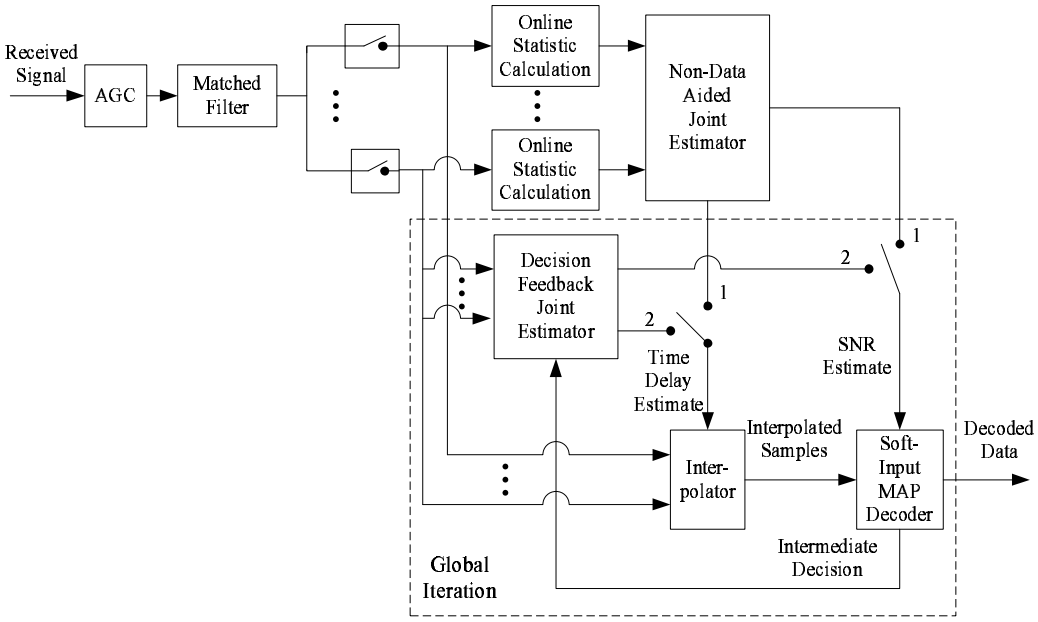


Figure 3.2: System model of joint symbol timing synchronization and SNR estimation in AWGN channels.

decision-aided methods. In order to do this, intermediate decoding results are used as a reference to the transmitted code word. The turbo decoder runs a few decoding iterations (local iterations) and then feeds intermediate decisions back to a decision feedback estimator, forming a global iteration. The switches flip to position 2. The new timing estimate is used to re-interpolate the samples, reconstructing a new input to the turbo decoder. The turbo decoder reuses the extrinsic information from previous local iterations, but uses the new reconstructed samples and SNR estimates for further decoding. The decision feedback estimation requires knowledge of the energy transmitted per symbol E_s (if the code rate is r, then the energy per information bit E_b is E_s/r), therefore an automatic gain control (AGC) block is needed. We assume that perfect AGC is available so that E_s is normalized to unity.

Assuming that (root) raised cosine-rolloff (RC-rolloff) pulse shaping is used, the output

of the matched filter is

$$r(t) = \sum_{k=-\infty}^{\infty} \sqrt{E_s} a_k g(t - kT) + w(t) \tag{3.4}$$

where $\{d_k\}$ is the transmitted code sequence, which for BPSK is $a_k \in \{-1, +1\}$, $g(t)$ is the RC-rolloff pulse shape function with a rolloff factor α [1], and $w(t)$ is additive noise. $r(t)$ is sampled N times per symbol period, and the kth sample taken by the nth sampler is

$$r_n[k] = r\left(kT + \frac{n-1}{N}T - \frac{T}{2} + \tau\right) \quad 1 \le n \le N. \tag{3.5}$$

where τ is the timing offset. The assumption of perfect frame synchronization implies that $-T/2N \le \tau \le T/2N$. With perfect timing (i.e. $\tau = 0$) only one sample per symbol is needed (i.e. $N = 1$) and the output of the matched filter has no intersymbol interference (ISI). Thus the SNR of the samples is exactly E_s/N_0. However, when perfect timing is not available, the performance may degrade significantly. For RC-rolloff pulse shaping, the performance degradation is not only due to the loss in received signal power, but also due to the presence of rather severe ISI.

3.4 Effective SNR

As shown in Fig. 3 of Chapter 1 , improper timing results in signal energy loss and ISI. If $r(t)$ is sampled once each symbol period, then

$$r[k] = r(kT) = \sqrt{E_s} d_k g(\tau) + \sqrt{E_s} \sum_{i \ne k} d_i g(kT - iT + \tau) + w_k. \tag{3.6}$$

On the right side of (3.6), only the first time contains the required data, modified by $g(\tau)$. The second term in (3.6) represents interference from adjacent symbols, namely ISI. If $\tau = 0$,

then $r_k = \sqrt{E_s} d_k + w_k$. Otherwise the decision on each symbol is affected by symbol energy loss and possible ISI when pulse shaping is applied. Thus symbol error rate may increases due to improper timing. The task of a time synchronizer is to estimate τ and compensate for it.

We define effective SNR in comparison with channel SNR. The effective SNR is a function of both the channel SNR and the timing offset. It indicates the SNR loss due to time delay. When the received signal is free of noise, the error caused by non-perfect timing is $r(kT) - \sqrt{E_s} d_k$, where $w(t) = 0$. The timing error results in interference from adjacent symbols and a loss in received signal energy. It is characterized by the normalized mean squared error (MSE) [48]. When a RC pulse is used for pulse shaping and timing offset $\tau \neq 0$, this MSE is

$$M(\tau) = \sum_{k=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} m_{k-j} g(\tau - kT) g(\tau - jT) - 2 \sum_{k=-\infty}^{\infty} m_k g(\tau - kT) + m_0, \quad (3.7)$$

where m_k is the autocorrelation of the coded sequence $\{x_k\}$. Assuming that the x_k's are independent and zero-mean, the autocorrelation of the coded sequence is $m_k = 1$ if $k = 0$, and zero elsewhere. Therefore, the mean squared error is

$$M(\tau) = \sum_{k=-\infty}^{\infty} g^2(\tau - kT) - 2g(\tau) + 1. \quad (3.8)$$

Note that this is an even function of τ.

Because the ISI is independent of the channel noise, it could be modeled as an additional Gaussian noise component [13]. More specifically, we define the effective SNR β as the SNR at a particular timing offset when all the effects are counted including the additive noise, the ISI, and the loss of signal power. It can be expressed as a function of both the channel

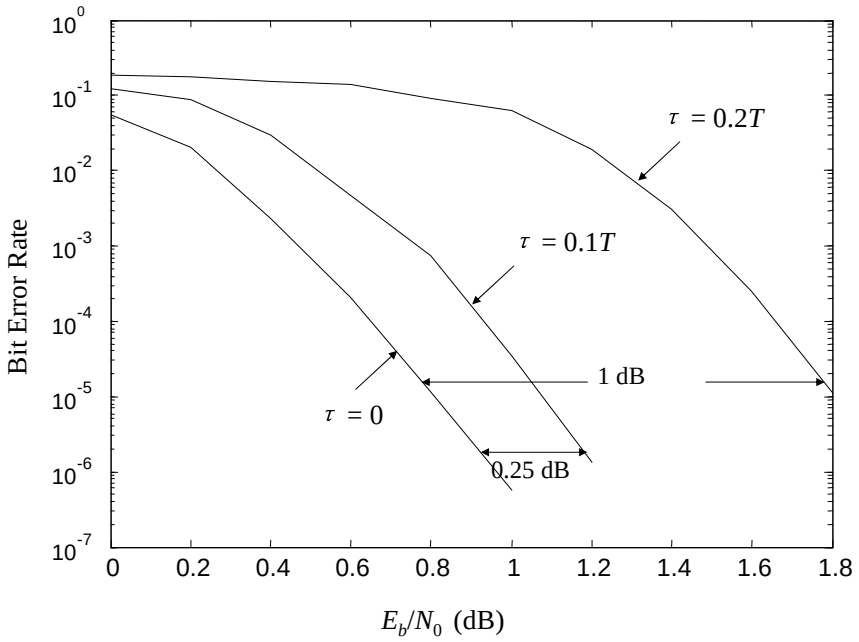


Figure 3.3: BER performance of a rate 1/3 turbo code with fixed timing offset. Interleaver size = 1530. Constituent convolutional codes have constraint length = 4.

SNR β_0 and the timing offset τ.

$$\beta\left(\tau, \beta_0\right) \quad = \quad \frac{g(\tau) E_s/N_0}{2M\left(\tau\right) E_s/N_0 + 1} \tag{3.9}$$

where $\beta_0 = \beta\left(0, E_s/N_0\right) = E_s/N_0$.

Fig. 3.3 shows the BER performance of a turbo code with fixed timing offset. The code uses the structure as defined in cdma2000 standard [49]. The coding rate is 1/3, and constituent RSC code is (15, 13). The interleaver size is 1530. BPSK modulation is used with raised-cosine pulse shape, where $\alpha = 0.5$. The decoder uses log-MAP [30] with 10 iterations. The coding gain loss for $\tau = 0.1T$ is about 0.25 dB at a BER of 10^{-5}, and over 1 dB for $\tau = 0.2T$. This loss illustrates the sensitivity of turbo codes to timing synchronization errors. This coding gain loss is also the difference between the effective SNR and the actual channel SNR at the given timing offset. Thus the effective SNR is a good way to characterize

the loss in coding gain.

3.5 Feedforward estimation

3.5.1 Online statistics

Since multiple samples per symbol are taken, they can be used to first estimate the effective SNR and then solve (3.9) for τ and β_0. If a set of $N \geq 2$ effective SNR values and the timing differences between them are known, an equation array can be established using (3.9). Now with $N \geq 2$ equations and two unknowns, it is possible to jointly determine β_0 and τ. Interpolation can help to recover the loss in signal energy when there are multiple samples available.

Successful implementation of the joint estimation strategy requires fairly accurate estimates of the effective SNR for each of the N sample positions averaged over the entire frame. To compute the effective SNR estimate, we use the approach proposed by Summers and Wilson [50] which computes online statistics using sample means of r_n^2 and $|r_n|$, i.e.

$$\hat{s}_n = \frac{1}{K}\sum_{k=1}^{K} r_n^2[k] \Bigg/ \left[\frac{1}{K}\sum_{k=1}^{K} |r_n[k]|\right]^2 \tag{3.10}$$

where K is the number of symbols in a frame. The online statistics are of interest because they are directly related to SNR and do not require knowledge about data. As K approaches infinity, the sample means become expected values,

$$s_n = \frac{\mathbf{E}[r_n^2]}{\{\mathbf{E}[|r_n|]\}^2} = \frac{1+2\beta_n}{\left[\sqrt{\frac{2}{\pi}}e^{-\beta_n} + \sqrt{2\beta_n}\mathrm{erf}\left(\sqrt{\beta_n}\right)\right]^2} = f(\beta_n) \tag{3.11}$$

where $\beta_n = \beta\left(\frac{n-1}{N}T - \frac{T}{2} + \tau, \beta_0\right)$, $1 \leq n \leq N$. Since β_n is a function of β_0 and τ, s_n is also

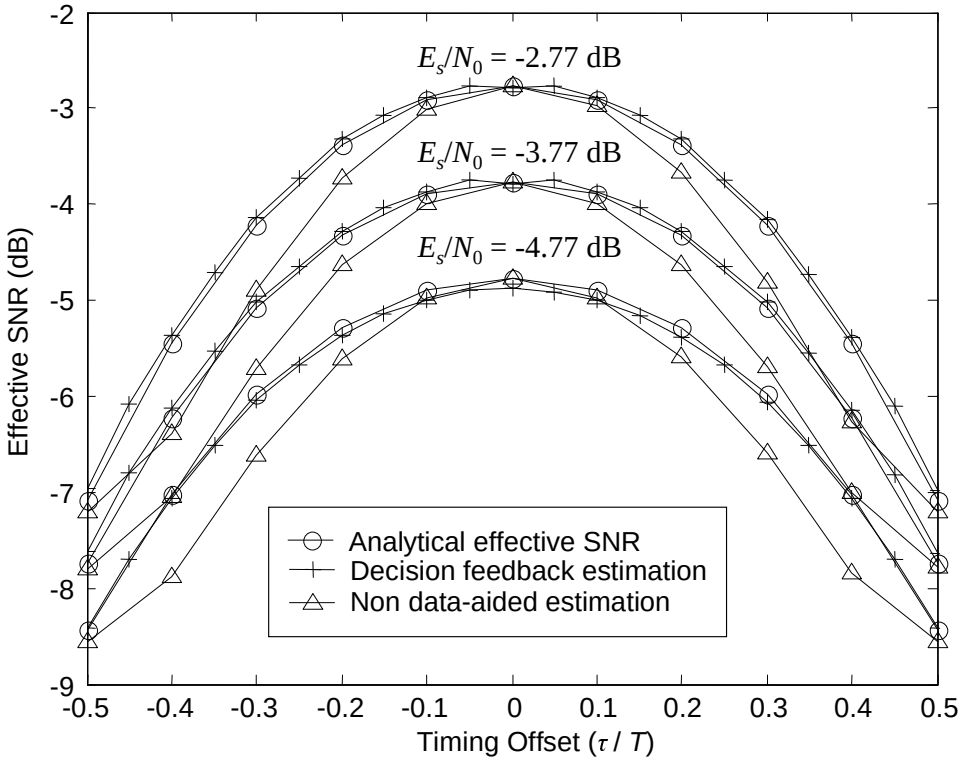


Figure 3.4: Effective SNR in the presence of improper timing, both using the non-data aided estimation and decision-feedback methods (100 trials with a frame size 4590). RC-rolloff pulse shaping is used with rolloff factor $\alpha = 0.5$.

related to β_0 and τ. The following terminologies are used now and hereafter. Let x be a parameter, for example τ, then x denotes the actual value, while $\hat{x}$ is an estimate of x.

Fig. 3.4 shows the results of a simulation that compares the estimated and analytical effective SNRs. Analytical effective SNRs are calculated from (3.9). Non-decision-aided (NDA) estimates of effective SNRs are obtained by calculating $\{s_n\}$ and inverting (3.11). In Fig. 3.4 the NDA estimated effective SNR fits the analytical results well for $|\tau| \leq 0.1T$. The NDA estimation produces pessimistic results of effective SNRs when $0.1T < |\tau| < 0.5T$ because, as discussed shortly, the online statistic generates a biased estimator of the effective SNR.

The received sequence $r_n[k], 1 \leq k \leq K$ is a discrete random process, therefore online statistics calculated by (3.10) are random variables. Before the synchronizer can be designed

and evaluated, the online statistics must first be characterized. Detailed derivations are shown in Appendix E. The expected value of $\hat{s}_n$ is

$$\begin{aligned}\mathbf{E}\left[\hat{s}_n\right] &= (1+2\beta_n)\left[\frac{1}{\sqrt{\frac{2}{\pi}}\exp\left(-\beta_n\right)+\sqrt{2\beta_n}\mathrm{erf}\left(\sqrt{\beta_n}\right)}\right]^2+\frac{\left[g\left(\frac{n-1}{N}T-\frac{T}{2}+\tau\right)E_s\right]^2}{K} \\ &\cdot \left(1+\frac{1}{2\beta_n}\right)\left(1+\frac{1}{2\beta_n}-\left[\sqrt{\frac{2}{\pi}}\exp\left(-\beta_n\right)+\sqrt{2\beta_n}\mathrm{erf}\left(\sqrt{\beta_n}\right)\right]^2\right) \qquad (3.12)\end{aligned}$$

The first term in (3.12) is exactly the same as the right-hand side of (3.11). However, the second term in (3.12) is greater than zero, thus the online statistics are inevitably biased. Since the online statistic s_n is a decreasing function of β_n, the estimated effective SNR is always less than the real value. If either K or β_n is large, then the second term is negligible and $\hat{s}_n$ is approximately an unbiased estimator. The variance of the online statistic is

$$\begin{aligned}\mathbf{V}\left[\hat{s}_n\right] &= \left[\frac{2\sigma^4}{K}+\frac{4\sigma^2}{K}+\left(\sigma^2+E_s\right)^2\right]\left(4\sigma_v^2+2\sigma_v^4\right) \\ &+ \left(\frac{2\sigma^4}{K}+\frac{4\sigma^2}{K}\right)\left\{\left[\frac{1}{\sigma\sqrt{\frac{2}{\pi}}\exp\left(-\frac{E_s}{2\sigma^2}\right)+\sqrt{E_s}\mathrm{erf}\left(\sqrt{\frac{E_s}{2\sigma^2}}\right)}\right]^2+\sigma_v^2\right\}. \qquad (3.13)\end{aligned}$$

where σ_v^2 is the variance of $\frac{1}{K}\sum_{k=1}^{K}\left|r_n\left[k\right]\right|$. $\mathbf{V}\left[\hat{s}_n\right]$ decreases when K grows. Intuitively, this is reasonable because an estimate becomes more reliable with a larger observation window.

3.5.2 MMSE algorithm

The estimation problem is to find the timing offset τ and SNR β_0 of the channel based on multiple sequences of samples obtained from the matched filter. Online statistics are calculated using these sample sequences. The relationship between $\{s_n\}$ and the pair of

parameters τ and β_0 is

$$s_n = s\left(-\frac{T}{2} + \frac{n-1}{N}T + \tau, \beta_0\right) \tag{3.14}$$

where $s(\tau, \beta_0) = f[\beta(\tau, \beta_0)]$. Intuitively, the optimal MMSE solution to this problem would result in a curve-fitting algorithm. The idea is to find the effective SNR curve with a certain pair of τ and β_0 that best fits the existing series of $\{\hat{s}_n\}$. An example is shown in Fig. 3.5 for $N = 4$ samples per symbol when $\tau = 0.13T$ and $\beta_0 = -3.77$ dB. The error is defined as the Euclidean distance of the candidate curve to the values in $\{\hat{s}_n\}$. The goal is to pick $\hat{\tau}$ and $\hat{\beta}_0$ that minimizes the following error function

$$e\left(\hat{\tau}, \hat{\beta}_0\right) = \sum_{n=1}^{N}\left\{s\left(-\frac{T}{2} + \frac{n-1}{N}T + \hat{\tau}, \hat{\beta}_0\right) - \hat{s}_n\right\}^2 \tag{3.15}$$

The corresponding optimization equations are

$$\begin{aligned} \frac{\partial e\left(\hat{\tau}, \hat{\beta}_0\right)}{\partial \hat{\beta}_0} &= 0 \\ \frac{\partial e\left(\hat{\tau}, \hat{\beta}_0\right)}{\partial \hat{\tau}} &= 0 \end{aligned} \tag{3.16}$$

Direct solution of (3.16) is difficult. Alternatives are to solve the problem numerically, or to make assumptions that simplify it. The numerical approach requires a large library of entries to be stored in look-up tables, and is sensitive to noise. Hence we seek an approach to simplify the problem by applying a linearized approximation to the relationship of effective SNR and the pair of parameters (τ, β_0) as shown in Fig. 3.5. Since the online statistics $\{\hat{s}_n\}$ are directly available without needing to compute each $\hat{\beta}_n$, and uniquely map to the effective SNR as in (3.14), the algorithm can be directly applied to $\hat{s}_n$.

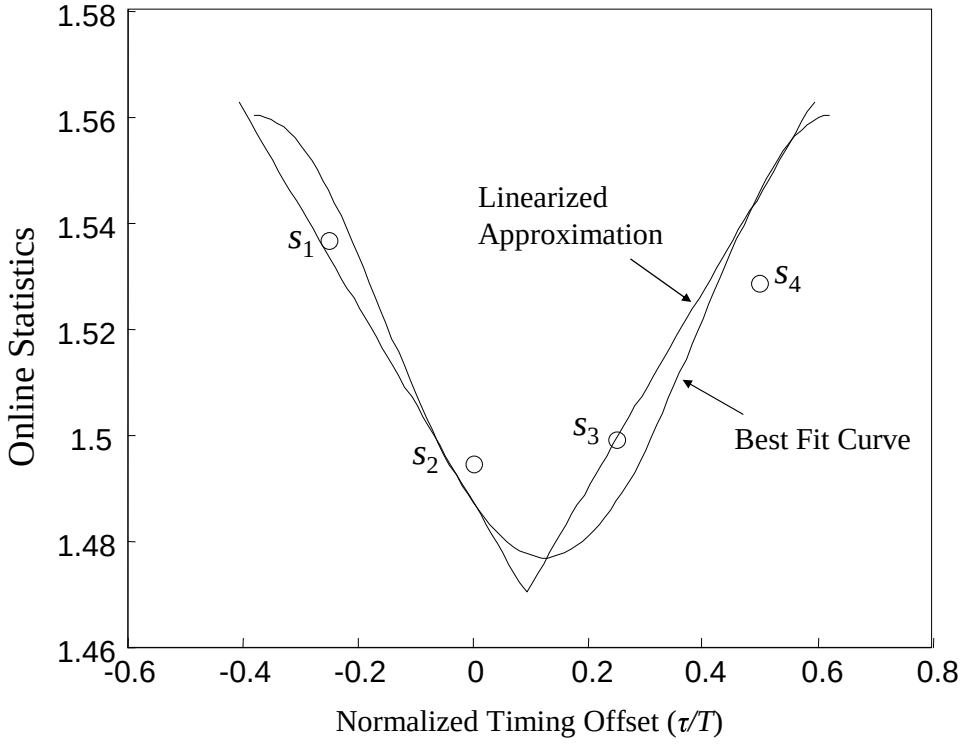


Figure 3.5: Example series of online statistics $\{\hat{s}_n\}$, the corresponding best fit curve, and a linearized approximation.

3.5.3 Reduced complexity estimation of β_0 and τ

Because the exact MMSE solution is complex and sensitive to noise, we have developed the following reduced complexity method for estimating β_0 and τ. An initial estimate of β_0 is found by selecting the minimum $\{\hat{s}_n\}$ and then inverting (3.11),

$$\hat{\beta}_0 = f^{-1}\left(\min_{1\le n\le N} \hat{s}_n\right) \tag{3.17}$$

For the curve fitting method, this estimate is used to select the initial candidate curve. In the linearized approximation $\hat{\beta}_0$ is used to determine the slope, i.e. the linearized approximation is

$$s\left(\hat{\tau}, \hat{\beta}_0\right) = C\left(\hat{\beta}_0\right)|\hat{\tau}| + \hat{s}_0 \tag{3.18}$$

where $C\left(\hat{\beta}_0\right)$ is the slope and $\hat{s}_0 = f\left(\hat{\beta}_0\right)$. Values of $C\left(\hat{\beta}_0\right)$ are found by evaluating the relationship between online statistics and τ. For a specific β_0,

$$C\left(\beta_0\right) = \frac{s\left(0.5T, \beta_0\right) - s_0}{0.5T} \tag{3.19}$$

Empirical values of $C\left(\beta_0\right)$ are found by simulations and are stored in a look-up table. The linearization approximation of $C\left(\beta_0\right)$ is $C\left(\beta_0\right) = 0.0269\beta_0 + 0.2133$, where β_0 is in dB.

The optimization function for $\hat{\tau}$ is

$$\frac{\partial e}{\partial \hat{\tau}} = \sum_{n=1}^{N} 2C\left(\hat{\beta}_0\right) \text{sign}\left(n - 1 - \frac{N}{2} + \hat{\tau}\right)\left[C\left(\hat{\beta}_0\right)\left|-\frac{T}{2} + \frac{n-1}{N}T + \hat{\tau}\right| + \hat{s}_0 - \hat{s}_n\right]. \tag{3.20}$$

$\hat{\tau}$ is found by setting (3.20) equal to zero. The slope $C\left(\hat{\beta}_0\right) \neq 0$, hence

$$\hat{\tau} = \frac{T}{2N} - \frac{1}{C\left(\hat{\beta}_0\right) N} \sum_{n=1}^{N} \left[\text{sign}\left(-\frac{T}{2} + \frac{n-1}{N} T \right) (\hat{s}_0 - s_n) \right]. \tag{3.21}$$

Under the linearized approximation, the true value of τ should satisfy

$$\tau = \frac{T}{2N} - \frac{1}{C\left(\beta_0\right) N} \sum_{n=1}^{N} \left[\text{sign}\left(-\frac{T}{2} + \frac{n-1}{N} T \right) (\hat{s}_0 - \bar{s}_n) \right], \tag{3.22}$$

where $\bar{s}_n = \mathbf{E}\left[\hat{s}_n\right]$. The mean square error of the timing estimate is found to be

$$\mathbf{E}\left[(\tau - \hat{\tau})^2\right] = \frac{1}{\left[C(\hat{\beta}_0)N\right]^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \rho_{ij} \text{sign}\left[(i - 1 - \frac{N}{2})(j - 1 - \frac{N}{2}) \right] \sqrt{\mathbf{V}\left[\hat{s}_i\right] \mathbf{V}\left[\hat{s}_j\right]}, \tag{3.23}$$

where ρ_{ij} is the correlation coefficient of s_i and s_j, $1 \leq i, j \leq N$. This coefficient is symmetric, $\rho_{ij} = \rho_{ji}$, and is periodic due to the fact that the online statistics count all the samples in a frame. When $\alpha = 0.5$, $\rho_{i,i+1} = 0.1657$, 0.3229 and 0.4792 when $N = 2$, 3 and 4, respectively.

The root-mean square (RMS) timing error with 2, 3, and 4 samples per symbol using the proposed joint estimation algorithm is shown in Fig. 3.6. The analytical curves are found by plotting (3.23). The simulated and analytical results agree, especially as the SNR and number of samples per symbol gets large.

3.5.4 Interpolation

Once the estimate of τ is available, the interpolator combines information from candidate samples to construct what the samples would have been, had the waveform been sampled at one sample per symbol at exactly the proper instant. This is equivalent to reconstructing the waveform and sampling it with proper timing. These samples are input to the turbo decoder.

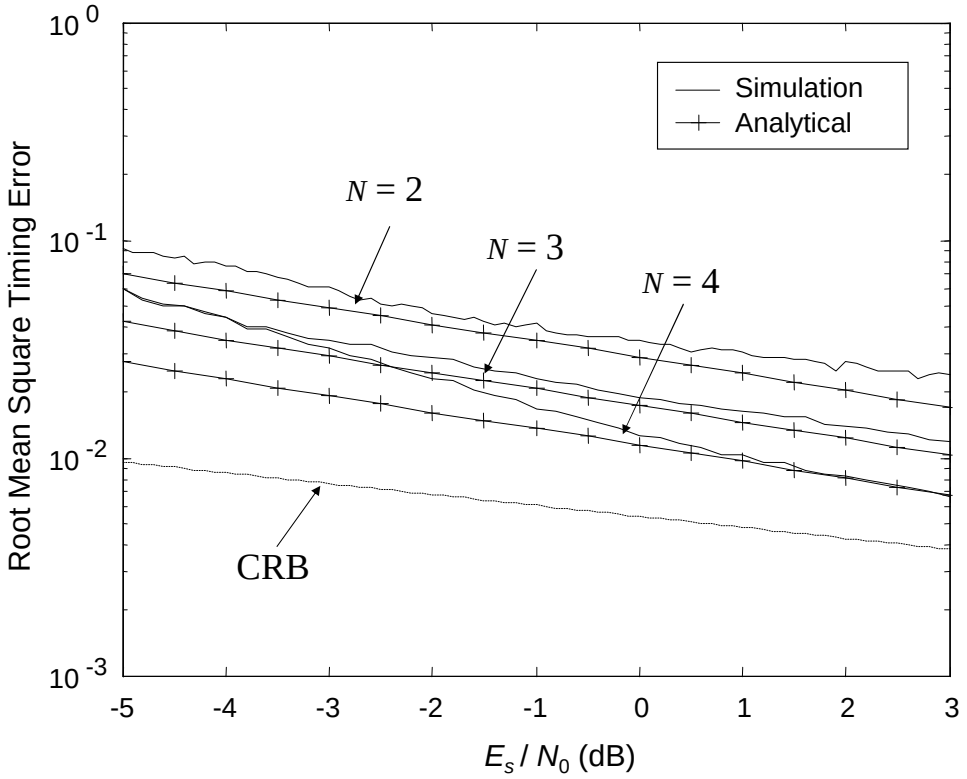


Figure 3.6: RMS timing error when $K = 4590$, with N equal to 2, 3, and 4, uncoded data sequence with BPSK modulation and rolloff factor $\alpha = 0.5$.

The simplest interpolator is a linear interpolator, which generates a weighted summation of selected samples. In the proposed strategy, just the two samples that are closest to the estimated timing offset are selected. The other samples, if available, are not used because they are subject to higher additive noise, and consequently are not as reliable. Assuming that the two samples come from the nth and $(n+1)$th samplers, then the sampling time associated with these samples are $\hat{\tau}_n = -\frac{T}{2} + \frac{n-1}{N}T + \hat{\tau}$ and $\hat{\tau}_{n+1} = \hat{\tau}_n + T/N$, respectively. The result of the linear interpolation is the following statistic for the kth symbol

$$r[k] = w_n r_n[k] + w_{n+1} r_{n+1}[k], \tag{3.24}$$

where the weights $w_n = \frac{|\hat{\tau}_{n+1}|}{|\hat{\tau}_n| + |\hat{\tau}_{n+1}|}$ and $w_{n+1} = \frac{|\hat{\tau}_n|}{|\hat{\tau}_{n+1}| + |\hat{\tau}_{n+1}|}$.

This simple interpolation method results in a signal energy loss. However, this loss is negligible when $N > 2$ from observation of simulation results. We have also tested higher order polynomial interpolators [51]. The performance of more complex interpolaters is usually worse than with using simple linear interpolation because these interpolaters are too sensitive to noise, so they are not suitable for the low SNR environment where turbo codes are implemented.

3.5.5 Final estimation of β_0

As shown in Fig. 3.5, the linearized approximation reaches its minimum value at

$$\hat{s}_0 = \frac{1}{N}\sum_{n=1}^{N}\left[\hat{s}_n + C\left(\hat{\beta}_0\right)\left|-\frac{T}{2} + \frac{n-1}{N}T + \hat{\tau}\right|\right]. \tag{3.25}$$

When $\hat{\tau}$ is available, a final estimate of β_0 is found

$$\hat{\beta}_0' = f^{-1}\left(\frac{1}{N}\sum_{n=1}^{N}\left[\hat{s}_n + C\left(\hat{\beta}_0\right)\left|-\frac{T}{2} + \frac{n-1}{N}T + \hat{\tau}\right|\right]\right). \tag{3.26}$$

This estimate is used in the turbo decoder to generate the channel reliability information. Turbo codes are not sensitive to moderate errors in the estimation of the SNR [50]. Our simulation also shows that the performance loss due to estimating β_0 is negligible.

3.6 Decision-directed refinery

3.6.1 Decision-aided (DA) SNR estimation

When the data is known (or can be accurately estimated), the variance of the additive noise can be estimated as proposed by Reed and Asenstorfer in [52]. The additive noise includes the effects of both the channel noise and the ISI. Let $\sigma^2 = N_0/2$ be the variance of channel noise and σ_n^2 the additive noise on the n-th sample sequence. Assuming that the transmitted data is available at the receiver, the variance of the additive noise is a function of (τ, β_0) and can be expressed as

$$v(\tau, \beta_0) = \mathbf{E}\left[\left\{r(kT+\tau) - \sqrt{E_s}a_k\right\}^2\right] = E_s\left\{[g(\tau)-1]^2 + M(\tau) + \frac{1}{2\beta_0}\right\}. \quad (3.27)$$

$v(\tau, \beta_0)$ is an even function of τ and $v(0, \beta_0) = \sigma^2$. The effective SNR is

$$\beta(\tau, \beta_0) = \frac{g(\tau) E_s}{2v(\tau, \beta_0)} \quad (3.28)$$

The ideal AGC block in the system model normalizes E_s. Therefore $E_s = 1$ in the following. The validity of (3.27) is verified by simulation as shown in Fig. 3.4. This simulation uses the estimated variance $\hat{\sigma}_n^2$ derived below and the actual data for decision feedback. When implemented in a turbo decoder, the feedback simply uses a hard decision of the log-likelihood ratio (LLR) $\hat{a}_k = \text{sign}(y[k])$, where $y[k]$ is the LLR of the kth information bit. We also considered soft-decision feedback by using $\hat{a}_k = \tanh(y[k]/2)$ but found essentially

no performance difference when compared with hard-decision feedback.

For limited length sequences, an estimate of σ_n^2 can be found by replacing the expected value with the sample mean

$$\hat{\sigma}_n^2 = \frac{1}{K}\sum_{k=1}^{K}\left[\left(r_n[k]-a_k\right)^2\right]. \tag{3.29}$$

The point estimator defined in (3.29) is an unbiased estimator for $v\left(\frac{n-1}{N}T+\frac{T}{2}+\tau,\beta_0\right)$. Its variance is

$$\mathbf{V}\left[\hat{\sigma}_n^2\right] = \frac{2}{K}\left[v\left(\frac{n-1}{N}T+\frac{T}{2}+\tau,\beta_0\right)\right]^2. \tag{3.30}$$

Similar to non-decision-aided estimation, we establish the following error function

$$e_{DA}(\tau,\beta_0) = \sum_{n=1}^{N}\left\{v\left(-\frac{T}{2}+\frac{n-1}{N}T+\hat{\tau},\hat{\beta}_0\right)-\hat{\sigma}_n^2\right\}^2, \tag{3.31}$$

which we wish to minimize with respect to $\hat{\tau}$ and $\hat{\beta}_0$.

3.6.2 Estimation of β_0 and τ

Since $v(\tau,\beta_0)\geq\sigma^2$, we select the estimate of σ^2 as $\hat{\sigma}^2=\min\limits_{1\leq n\leq N}\hat{\sigma}_n^2$. Thus the corresponding estimate of the channel SNR is

$$\hat{\beta}_{0,DA}=\frac{1}{2\hat{\sigma}^2}. \tag{3.32}$$

The linearized approximation of (3.27) is

$$v\left(\hat{\tau},\hat{\beta}_0\right)=C_{DA}\left(\hat{\beta}_0\right)|\hat{\tau}|+\hat{\sigma}^2 \tag{3.33}$$

In our simulations, all candidate $C_{DA}(\beta_0)$ are stored in a look-up table. The linearization approximation is $C_{DA}(\beta_0) = 0.0650\beta_0 + 0.8233$, where β_0 is in dB.

Similar to the process of (3.20) and (3.21), we find the following DA estimate of τ

$$\hat{\tau}_{DA} = \frac{T}{2N} - \frac{1}{C_{DA}\left(\hat{\beta}_{0,DA}\right) N} \sum_{n=1}^{N} \left[\text{sign}\left(-\frac{T}{2} + \frac{n-1}{N} T \right) \left(\hat{\sigma}^2 - \hat{\sigma}_n^2 \right) \right] \tag{3.34}$$

The mean square error of the timing estimate is

$$\begin{aligned} & \mathbf{E}\left[(\tau - \hat{\tau}_{DA})^2 \right] \\ = \; & \frac{1}{\left[C_{DA}(\hat{\beta}_0) N \right]^2} \sum_{i=1}^{N} \sum_{j=1}^{N} \rho'_{ij} \text{sign}\left[(i - 1 - \frac{N}{2})(j - 1 - \frac{N}{2}) \right] \sqrt{\mathbf{V}\left[\hat{\sigma}_i^2\right] \mathbf{V}\left[\hat{\sigma}_j^2\right]} \end{aligned} \tag{3.35}$$

When $\alpha = 0.5$, the correlation values $\rho'_{i,i+1} = 0.537$, 0.779 and 0.874 when $N = 2$, 3 and 4 respectively. These values were calculated using the sample mean and sample variances of the simulation results. The noise is correlated because the matched filter colors the additive noise. Therefore $w(t)$ has an auto-correlation function equal to $g(t)$. The root-mean-square timing estimation error is shown in Fig. 3.7. The analytical curves are found by plotting (3.35). It is found that with knowledge of the transmitted data, the estimation error is close to the CRB, but the performance is sensitive to the accuracy of linearized approximation. When $N = 2$, or when the channel SNR is high, the linearized approximation turns inaccurate.

3.7 Simulations

A simulation campaign was completed in order to illustrate the effectiveness of the proposed estimation techniques. In the simulations, BPSK modulation was used over an AWGN channel. The timing error was quasi-static in the sense that the offset was constant throughout

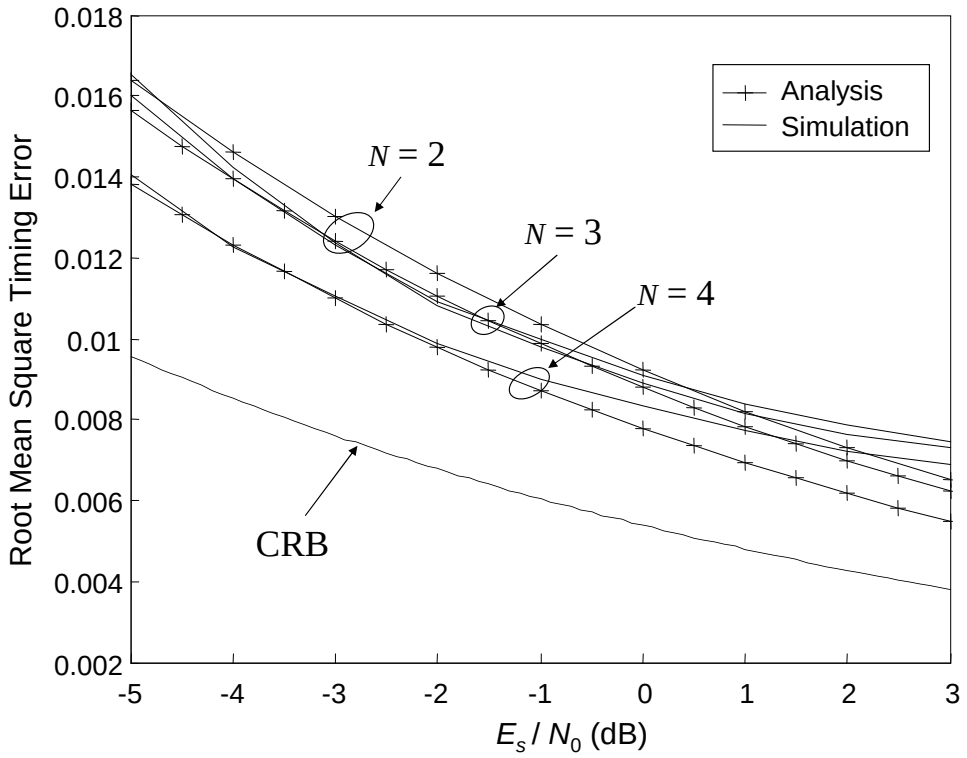


Figure 3.7: Root-mean square error of the decision-feedback timing estimate for uncoded BPSK modulation with rolloff factor $\alpha = 0.5$, frame size $K = 4590$, and $N = \{2, 3, 4\}$ samples per symbol.

the frame but varied independently from frame-to-frame according to a uniform distribution. The timing offset was estimated using the linearized approximation, and a lookup table with 31 candidate slope values was used, with E_s/N_0 ranging from -4.8 dB to -1.8 dB in 0.1 dB increments. This range corresponds to E_b/N_0 between 0 dB and 3 dB when the coding rate is 1/3. This range was selected because this is where the BER performance of typical turbo codes changes most rapidly, i.e. the range contains the so-called "waterfall". Turbo decoding is performed using the log-MAP algorithm [30].

We define two types of iteration, *local* and *global*. A local iteration is merely an iteration within the turbo decoder. On the other hand, a global iteration is an iteration between the turbo decoder and the estimator. The number of times the SNR and timing offset are estimated is equal to the number of global iteration. In all simulations, the total number of local iterations is set to 10. Systems that do not use decision feedback only execute one global iteration, while systems with decision feedback execute two global iterations (with five local iterations per global iteration). The balance between the number of global and local iterations is important. On the one hand, decision-directed estimation works better when the intermediate decision is accurate, and thus it is important to not feed back information prematurely. On the other hand, it is important not to wait too long before feeding back information to the estimator or else the benefits of the decision-feedback estimator will not be realized.

As in [13], the first turbo code tested uses a 256 bit random interleaver. The constituent RSC code uses octal generators (37, 33), and the parity bits are alternatively punctured to obtain a coding rate of 1/2. The two constituent codes are terminated independently. The BER performance is shown in Fig. 3.8 for $N = 2, 3$, and 4 samples per symbol and both the non-decision-aided and the decision-directed estimation techniques. The performance with perfect timing and SNR estimates is also shown for comparison purposes. With only two samples per symbol and non-decision-aided estimation, the BER performance is within 0.8

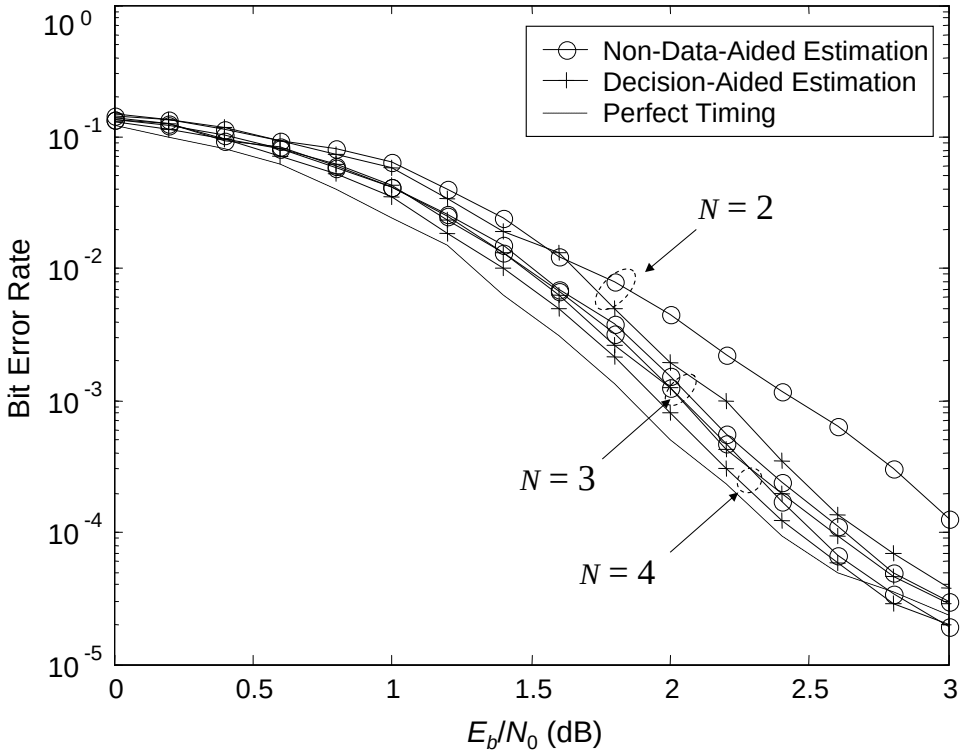


Figure 3.8: Performance of turbo code one using the proposed joint timing/SNR algorithms and $N = \{2, 3, 4\}$ samples per symbol. The interleaver size is 256, overall code rate is 1/2, RSC generators (37, 33), and decoding uses 10 total iterations of log-MAP algorithm.

dB of perfect timing at a BER of 10^{-4}. If a second global iteration is used (decision-feedback estimation), then this loss is reduced to only about 0.4 dB. With four samples per symbol and non-decision-aided estimation, performance is within 0.2 dB of perfect timing at a BER of 10^{-4}, and with decision-feedback estimation the loss is less than 0.1 dB.

The second code that we consider is one of the turbo codes defined in the cdma2000 standard [49]. In particular, an interleaver size of 1530 and overall code rate of 1/3 is selected, which will make this code significantly stronger than the first code. This code uses RSC constituent codes with octal generator $(15, 13)$ and interleaver as defined in the standard. The BER performance is shown in Fig. 3.9, again for $N = 2, 3$, and 4 samples per symbol and both estimation techniques. The BER losses in this case are very similar

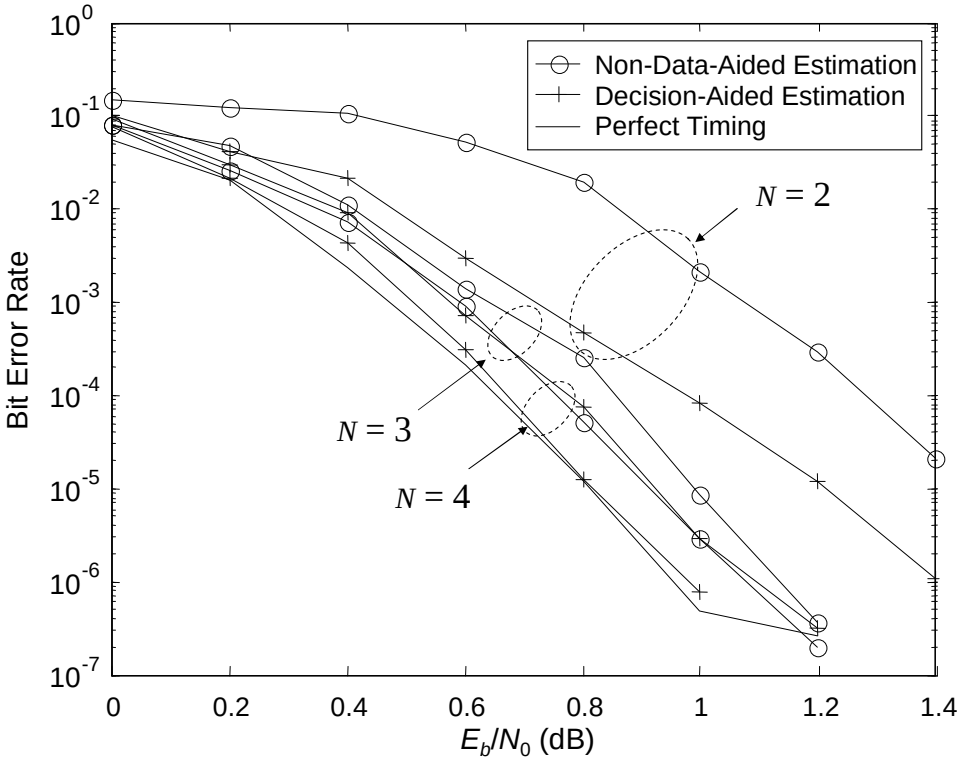


Figure 3.9: Performance of turbo code two using the proposed joint timing/SNR algorithms and $N = \{2, 3, 4\}$ samples per symbol. The code is as specified in the cdma2000 standard with overall code rate 1/3 and interleaver size 1530. Decoding uses 10 total iterations of log-MAP algorithm.

to the losses observed with the first code, indicating that the estimation technique is robust enough to work with stronger codes and at the corresponding low SNRs. In particular, with $N = 2$ and at a BER of 10^{-4} the loss relative to perfect timing with the non-decision-aided estimator is 0.8 dB and with the decision-directed estimator is about 0.4 dB. With $N = 4$ these two losses are 0.2 dB and 0.1 dB, respectively. In the above systems, it is shown that the BER performance improves as N increases if perfect timing is not available. Data-directed estimation helps to close the gap, thereby approaching the BER performance with perfect timing.

In Fig. 3.10, we compare the performance of both codes using our proposed estimation techniques with the previously proposed soft-combining method of [13] with 4 samples per

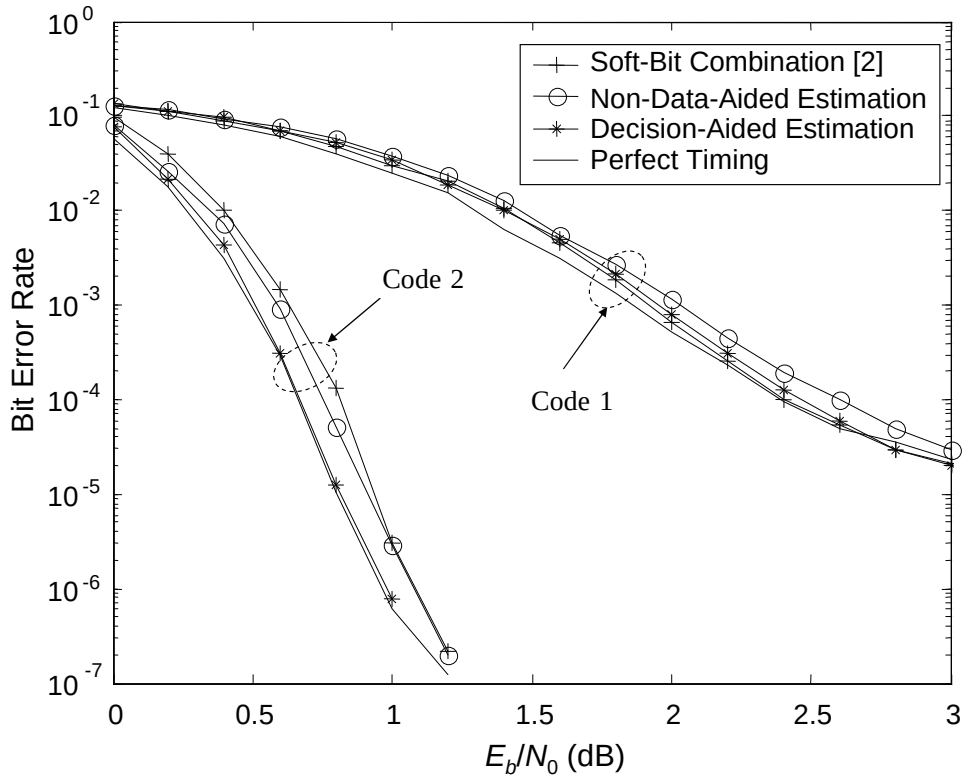


Figure 3.10: Comparison of Mielczarek's soft-combining method [13] and joint estimation of SNR and timing offset method with 4 samples per symbol. Code 1: interleaver size = 256, coding rate = 1/2, constituent RSC code is (37, 33); Code 2: interleaver size = 1530, coding rate = 1/3, constituent RSC code is (15, 13). Random interleaver is used and 10 iteration.

symbol. While soft-bit combining works better with the weaker code, the proposed joint estimation techniques outperform it when using the stronger code. This is due to the fact that the estimation error of the proposed frame-based estimator is inversely proportional to the window (frame) size. Recall, however, that the method in [13] requires two turbo decoders running in parallel, while our method only requires a single turbo decoder. The overall system complexity of the proposed approach is therefore significantly less than the one proposed in [13].

We also considered a third code, which is much stronger than the first two. This code is also defined in the cdma2000 standard [49]. The interleaver size is 20730. The code can

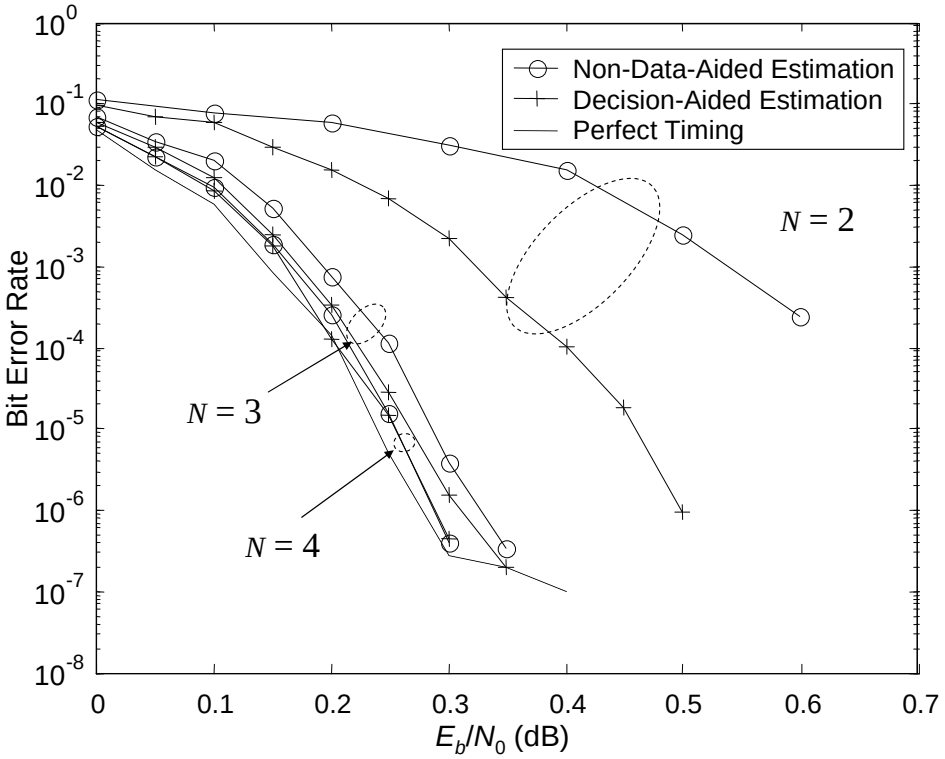


Figure 3.11: Performance of turbo code three using the proposed joint timing/SNR algorithms and $N = \{2, 3, 4\}$ samples per symbol. The code is as specified in the cdma2000 standard with overall code rate 1/3 and interleaver size 20730. Decoding uses 10 total iterations of log-MAP algorithm.

get a BER of 10^{-5} at a SNR of 0.23 dB with perfect timing. With only 2 samples per symbol and non-decision-aided estimation, the coding gain loss is within 0.8 dB of the BER performance for perfect timing at a BER of 10^{-4}. When $N = 2$ and one global iteration is invoked, the coding gain loss is about 0.3 dB at a BER of 10^{-5}. With 4 samples per symbol and non-decision-aided estimation, the coding gain loss is within 0.05 dB of the BER for perfect timing at a BER of 10^{-5}. When one global iteration is called, the coding gain loss does not improve much because of the low SNR.

In all the above simulations, it is found that the performance of $N = 3$ with global iteration is comparable to that of $N = 4$ without global iteration. The performance gain is achieved at the cost of one call of decision-aided joint estimation and re-interpolation. This

means we can design a system with a lower sampling rate and additional system complexity to achieve the similar performance with higher sampling rate and lower complexity.

3.8 Conclusion

Imperfect timing causes a loss in effective SNR which results in a severe BER performance degradation for timing shifts greater than about 10% of the symbol period. This performance loss can be recovered by a proper estimation algorithm. However, the situation is complicated by the fact that the channel SNR over which turbo codes operate is both very small and not known to the receiver. Also, practical systems are likely to use Nyquist pulse shaping, which introduces ISI in the presence of timing errors. Our approach to synchronization and SNR estimation involves sampling the signal multiple times per symbol period and computing an online statistic for each of the sample instances over the entire frame. These online statistics are then used to simultaneously estimate the channel SNR and timing offset. A simple linear interpolation algorithm is then used to reconstruct the matched filter samples at the estimated timing instants. Tentative decisions from the decoder can be fed back and used to refine the timing and SNR estimates.

The proposed algorithm recovers much of the loss due to poor synchronization, and does so with negligible added complexity and latency (compared to that of the turbo decoding algorithm itself). With feedback from the decoder to the estimator, the simulated coding gain loss is negligible, i.e. about 0.1 dB with 4 samples per symbol when the interleaver size = 1530 at a BER of 10^{-5}. Finally, it is noted that the proposed technique is suitable for more than just turbo codes. Indeed, any system can use the non-decision-aided technique. Furthermore, since only hard-decisions were fed back from the decoder to the estimator, the proposed iterative synchronization strategy is suitable for any error control code, not just those that use soft-output decoders.

Chapter 4

Mitigation of Random Time Walk

This chapter investigates the problem of timing recovery using fractional multiple samples per symbol. Usually the sampling rate is assumed to be a strict integer multiple of the symbol rate. A more practical assumption is that the multiple is a real number very close to a known integer but has an unknown fractional part. The sampler may also have jitter which makes the sampling time drift in a random manner. This chapter studies the effect of random time walk on turbo coded systems in very low signal-to-noise ratio environments. The random time walk is mitigated by overlapped sliding window and decision-feedback timing recovery.

4.1 Introduction

The concept of symbol timing recovery using oversampling and interpolation is becoming attractive thanks to availability of high-speed analog-to-digital converters and tremendous computational power. Usually the received signal is sampled two to four times per symbol. A timing error detector (TED) estimates the time delay between the actual and ideal sampling time [8][9][10][11][12][13][14].

The authors of references [8][9][10][11][12] considered feedforward-only non-data-aided (NDA) estimation for random sequences. Hence the techniques introduced therein are applicable to general system schemes, including coded systems. The precision of timing estimators is proportional to the length of observation windows as indicated by the lower bounds [11][47]. When the data sequence is coded, especially turbo coded, the system is required to operate at signal-to-noise ratios (SNR) so low that very long observation windows are required to achieve desirable accuracy of timing estimation [13][14].

All the above publications assume that the time delay is changing slowly so that it is constant during the observation interval. They also assume that the sampling rate is a known integer multiple of the symbol rate. However, the sampling time may drift randomly due to improper alignment and jitter in the local clocks at the transmitter and the receiver. Therefore the sampling time is no longer fixed. It wanders with random time walk. There are two types of random time walk. The first type, the *linear walk*, is a result of the fractional sampling rate. The second type, the *random jitter*, is characterized by the random drifting of the sampling time.

Barry *et al* [53] discussed the effect of the random jitter in magnetic storage systems using powerful convolutional codes. The system is assumed to have zero initial timing offset. The signal is sampled at the symbol rate. Decision-feedback timing error detector (DF/TED) using Mueller and Müller's method [5] is implemented. A turbo equalizer is used to remove remaining intersymbol interference.

This chapter studies the problem of random walk introduced by both fractional sampling rate and random drifting. A powerful turbo coded system is considered, therefore very low SNR values are treated. The application considered is a turbo coded wireless communication system with moderate code rate and moderate data rate in AWGN channels. The negative effect of time walk is mitigated by overlapped sliding windows and iterative processing.

When time walk exists, the feedforward timing estimate in an observation window may

only be applicable to the symbols close to the center of the observation window. A direct solution is to use overlapped sliding window. This trades computational complexity for higher accuracy.

An important feature of turbo codes is that the decoding process is inherently iterative. After one or more decoding iterations, intermediate decoding results are available to enable decision-feedback estimation. Usually the decision-feedback estimation generates a more precise estimate of the timing estimate.

The remainder of the chapter is organized as follows. Section 4.2 introduces the system model of the receiver. Section 4.3 analyzes the effect of random time walk. Timing recovery techniques are proposed in Section 4.4 to combat random time walk. Section 4.5 presents simulation results. Section 4.6 concludes the research work.

The following nominating rule is used in the chapter to present continuous and discrete signals. $x(t)$ is a continuous function, while $x[n]$ is a discrete sequence in the time domain. $j = \sqrt{-1}$ represents the imaginary unit.

4.2 System model

The system model of a receiver which implements timing recovery and turbo decoding is shown in Fig. 4.1. Without loss of generality, it is assumed that the time walk is introduced in the receiver. The received signal from an additive white Gaussian noise (AWGN) channel is

$$y(t) = \sqrt{E_s} \sum_{k=-\infty}^{\infty} d_k g_T(t - kT + \tau_0) + w(t), \tag{4.1}$$

where τ_0 is the relative time offset between the transmitter and receiver. Assuming perfect frame synchronization, τ_0 is a random variable with uniform distribution in the range $-T/2 \leq$

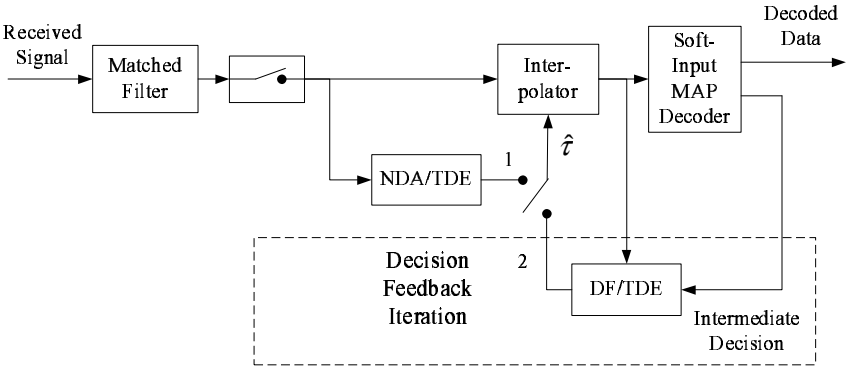


Figure 4.1: Receiver diagram using over-sampling and interpolation timing recovery.

$\tau_0 < T/2$. τ_0 is assumed to be constant during one frame. $w(t)$ is additive white Gaussian noise. The matched filter has an impulse response of $g_R(t) = g_T(T-t)$. The matched filter output is

$$r\left(t\right) = \sqrt{E_s} \sum_{k=-\infty}^{\infty} d_k g\left(t - kT + \tau_0\right) + w_M\left(t + \tau_0\right). \tag{4.2}$$

$w_M(t)$ becomes colored because of the receive filter.

$r\left(t\right)$ is sampled multiple times each symbol duration. The desirable sampling rate is an integer N, for example, two or four samples per symbol [8][10]. However, the actual sampling rate is $N_f = \frac{N}{1+\tau_1/T}$, where τ_1 is the fixed time delay that happens in one symbol period. There is also random jitter that comes from unexpected variations of the local clocks in the transmitter and the receiver. Let ξ_i represent the time jitter in the i-th symbol period. ξ_i is assumed to be an independent identically distributed (i.i.d.) Gaussian random variable with distribution $\mathcal{N}(0, \sigma_t^2)$. The random jitter in the k-th symbol $\tau_2[k] = \sum_{i=k_0}^{k} \xi_i$ is the accumulation of ξ_i's of all symbols of interest with indices $k_0 \leq i \leq k$, where k_0 is the starting point of the observation window. For the k-th symbol, the total delay is

$\tau[k] = \tau_0 + k\tau_1 + \tau_2[k]$. The output of the sampler is

$$\begin{aligned} r[n] &= r\left(\frac{nT}{N_f} + \tau_2[n]\right) \\ &= \sqrt{E_s}\sum_{k=-\infty}^{\infty} d_k g\left(\frac{nT}{N} - kT + \tau_0 + \frac{n\tau_1}{N} + \tau_2[n]\right) + w_M\left(\frac{nT}{N} + \tau_0 + \frac{n\tau_1}{N} + \tau_2[n]\right) \\ &= \sqrt{E_s}\sum_{k=-\infty}^{\infty} d_k g\left(\frac{nT}{N} - kT + \tau[n]\right) + w_M\left(\frac{nT}{N} + \tau[n]\right). \end{aligned} \tag{4.3}$$

One feature of a turbo coded system is that the decoding process is iterative. While one round of maximum *a posteriori* (MAP) decoding is called a *local iteration*, one round of timing estimation, interpolation, and decoding process is termed a *global iteration*. At least one global iteration is invoked when a codeword is received and decoded. In the first round global iteration, the switch is placed in position 1. The NDA/TED block finds out an estimate $\hat{\tau}$ of τ using samples $r[n]$ in an observation window consisting of $2L_F + 1$ symbols. $\hat{\tau}$ is then used in the interpolator to reconstruct the signal, known as interpolation timing recovery (ITR) [14]. The reconstructed samples then enter the soft-input turbo decoder.

After one or more local iterations, decoding results are available to facilitate decision-feedback time delay estimation. To begin a round of decision-feedback iteration, the switch is turned to position 2. The DF/TED uses the intermediate decision results and interpolated samples to calculate the timing difference between the previously estimated timing and proper timing. The timing estimate is updated by the timing difference. The samples $r[n]$ are re-interpolated according to the refined estimate $\hat{\tau}$ to generate new input for the turbo decoder. The extrinsic information is reused in the remaining turbo decoding iterations.

In the following text, it is assumed that one codeword is transmitted as one frame with perfect frame synchronization. As a quasi-static channel, τ_0 is an unknown constant in the transmission of one frame and τ_0 changes from frame to frame. τ_1 is a fixed constant for all frames. For simplicity of analysis, the symbol duration T is normalized and therefore

omitted.

4.3 Random time walk

To reveal the effect of random time walk, the special case of transmitting a single tone, i.e., a sinusoidal signal, in a noise-free environment is analyzed. The results can be extended to general cases of random data with Nyquist pulse-shaping in AWGN channels. It is assumed that $\tau_1 \ll 1$, and $\sigma_t^2 \ll 1$. The received signal is

$$r_s(t) = \sin\left(\frac{2\pi}{T}t + \tau_0\right). \tag{4.4}$$

The sampled result is

$$r_s[n] = \sin\left(\frac{n\pi}{N} + \tau_0 + \frac{n\tau_1}{N} + \tau_2[n]\right). \tag{4.5}$$

The digital filter technique introduced in [8] is used, where a square-law operation is implemented.

$$x[n] = |r_s[n]|^2. \tag{4.6}$$

The frequency component at the symbol rate is decomposed using discrete Fourier transform (DFT)

$$X = \sum_{n=-NL}^{NL-1} x[n]\, e^{-j2\pi\frac{n}{N}}. \tag{4.7}$$

The time delay is transformed to the phase of X and is found by estimate the angle of X.

$$\hat{\tau} = -\frac{1}{2\pi} \arg(X). \tag{4.8}$$

The function $\arg(X)$ gets the angle of X, where the return value is between $-\pi$ and π. $\hat{\tau}$ is in the range of $[-\frac{1}{2}, \frac{1}{2})$. It is shown in [8] that the time delay estimate is unbiased when $\tau_1 = 0$ and $\sigma_t^2 = 0$.

4.3.1 Known linear walk without random jitter

If τ_1 is known, then (4.7) can be modified to

$$\tilde{X} = \sum_{n=-\lfloor N_f L_F \rfloor}^{\lceil N_f(L_F+1) \rceil - 1} x[n] \, e^{-j2\pi \frac{n}{N_f}}, \tag{4.9}$$

and the time delay estimate is

$$\check{\tau} = -\frac{1}{2\pi} \arg\left(\tilde{X}\right) \tag{4.10}$$

If $\sigma_t^2 = 0$, then $\check{\tau}$ is still an unbiased estimate of τ. It has the same property as $\hat{\tau}$ in (4.8).

4.3.2 Unknown linear walk and random jitter

When τ_1 is unknown, the DFT result is

$$\begin{aligned} X &= \sum_{n=-NL}^{N(L_F+1)-1} x_s[n] \, e^{-j2\pi \frac{n}{N}} \\ &= \sum_{l=-L_F}^{L_F} \sum_{n=0}^{N-1} x_s[n + lN] \, e^{-j2\pi \frac{n}{N}}. \end{aligned} \tag{4.11}$$

Substituting (4.5) and (4.6) into (4.11), we obtain

$$\begin{aligned} X &= \sum_{l=-L_F}^{L_F} \sum_{n=0}^{N-1} \sin^2 \left\{ \pi \left[\frac{n}{N} + \tau_0 + \left(l + \frac{n}{N} \right) \tau_1 + \tau_2 [l] \right] \right\} e^{-j2\pi \frac{nk}{N}} \\ &= \sum_{l=-L_F}^{L_F} X^l \end{aligned} \tag{4.12}$$

where

$$\begin{aligned} X^l &= \sum_{n=0}^{N-1} \sin^2 \left\{ \pi \left[\frac{n}{N} + \tau_0 + \left(l + \frac{n}{N} \right) \tau_1 + \tau_2 [l] \right] \right\} e^{-j2\pi \frac{n}{N}} \\ &= \sum_{n=0}^{N-1} \frac{1}{2} \left(1 - \cos \left\{ 2\pi \left[\frac{n}{N} + \tau_0 + \left(l + \frac{n}{N} \right) \tau_1 + \tau_2 [l] \right] \right\} \right) \left(\cos \frac{2\pi n}{N} - j \sin \frac{2\pi n}{N} \right) \\ &\approx -\frac{N}{4} \left\{ \cos \left[2\pi \left(\tau_0 + l\tau_1 + \tau_2 [l] \right) \right] + j \sin \left[2\pi \left(\tau_0 + l\tau_1 + \tau_2 [l] \right) \right] \right\}. \end{aligned} \tag{4.13}$$

The last approximation is valid because $\tau_1 \ll 1$. X^l is further decomposed to

$$X^l = -\frac{N}{4} \left(\cos 2\pi\tau_0 + j \sin 2\pi\tau_0 \right) - \frac{N}{4} \left(\Delta_c^l + j\Delta_s^l \right), \tag{4.14}$$

where

$$\Delta_c^l = 2 \sin \left\{ \pi \left(l\tau_1 + \tau_2 [l] \right) \right\} \sin \left\{ \pi \left(2\tau_0 + l\tau_1 + \tau_2 [l] \right) \right\} \tag{4.15}$$

and

$$\Delta_s^l = 2 \sin \left\{ \pi \left(l\tau_1 + \tau_2 [l] \right) \right\} \cos \left\{ \pi \left(2\tau_0 + l\tau_1 + \tau_2 [l] \right) \right\} \tag{4.16}$$

The estimate value is

$$\hat{\tau} = \frac{1}{2\pi} \arctan \left(\frac{\sum_{l=-L_F}^{L_F} \Delta_s^l + N \sin 2\pi\tau_0}{\sum_{l=-L_F}^{L_F} \Delta_c^l + N \cos 2\pi\tau_0} \right). \tag{4.17}$$

It is clear from (4.17) that the effect of random time walk resembles noise around the constellation of $(N \cos 2\pi\tau_0, N \sin 2\pi\tau_0)$ in the Cartesian coordinates. The estimation error ϵ in a noise-free environment has three parts

$$\epsilon = \gamma_1\tau_1 + \gamma_2 + L_F\gamma_3. \tag{4.18}$$

γ_1 and γ_2 are both functions of τ_0, τ_1, and τ_2. γ_3 is related to τ_2. The absolute value of estimation error caused by linear walk when $\tau_1 = 0.01$ and $\sigma_t^2 = 0$ is plotted in Fig. 4.2 against τ_0/T. The estimation error is periodic and is apparently a sinusoidal curve. The magnitude of the curve is limited by τ_1. Note that this estimation error is only applicable to the center of the observation window. Other symbols have greater errors because the accumulation of time walk. When $L_F\tau_1$ is close to or greater than 1, cycle slips happen which cause burst errors.

Fig. 4.3 illustrates the relationship between L_F and the mean square estimation error of NDA/TED, when $\sigma_t^2 = 3 \times 10^{-8}$. Sinusoidal signal is transmitted in a noiseless channel. The estimation error is averaged over τ_0 in $[-T/2, T/2)$. The mean square errors are average values from 10,000 trials of simulation. The estimation error grows when L_F increases. This is because the accumulated random jitter is the sum of Gaussian random variables which has a greater variance. It is shown that an error floor appears when $\sigma_t^2 \neq 0$. When $L_F\tau_1$ is close to or greater than one, the estimation error increases exponentially.

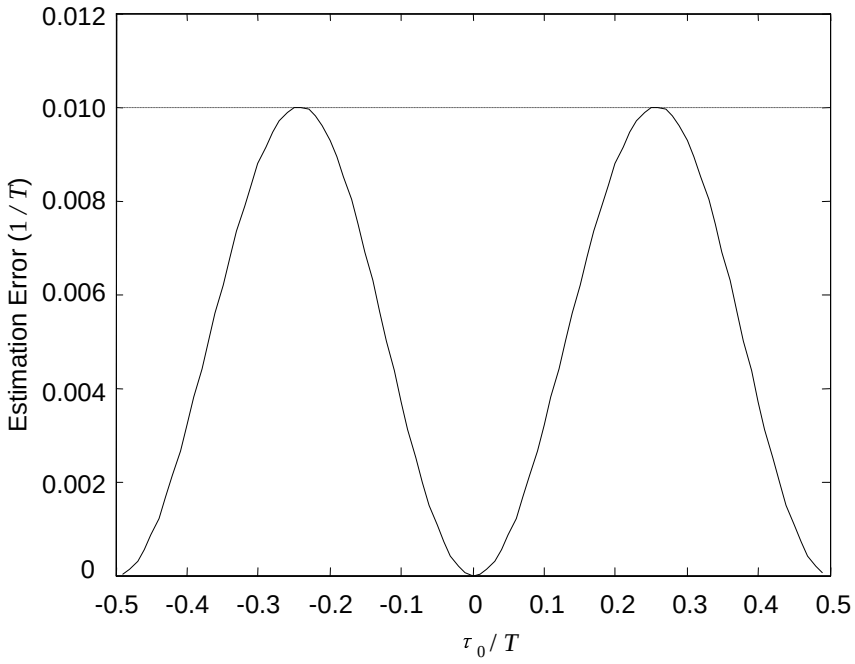


Figure 4.2: Timing estimation error due to unknown linear walk with $N = 4$, $\tau_1 = 0.01$, and $\sigma_t^2 = 0$. τ_1 is unknown to the receiver.

Fig. 4.4 depicts the performance of NDA/TED in terms of mean square estimation error in an additive white Gaussian noise (AWGN) channel with $\tau_1 = 10^{-3}$ and $\sigma_t^2 = 10^{-6}$. The curves are compared with the lower bound given by the Cramer-Rao bound [11]. Generally, the greater size the observation window has, the more precise estimate it produces. However, the error floor illustrated in Fig. 4.3 may prevent the estimate from converging no matter what the SNR is, as shown in Fig. 4.4. It is also shown that the TED with longer observation window suffers more from random jitter. There exists a trade-off between estimation precision and observation window size.

As previously mentioned, the symbols close to the boundaries of observation windows tend to have significantly greater errors than those close to the center of the windows. One straightforward solution is to use overlapped sliding windows. The estimation result in one observation window is only applied to the symbols in the middle of that window. The observation window moves along the frame to cover other symbols. Higher computational

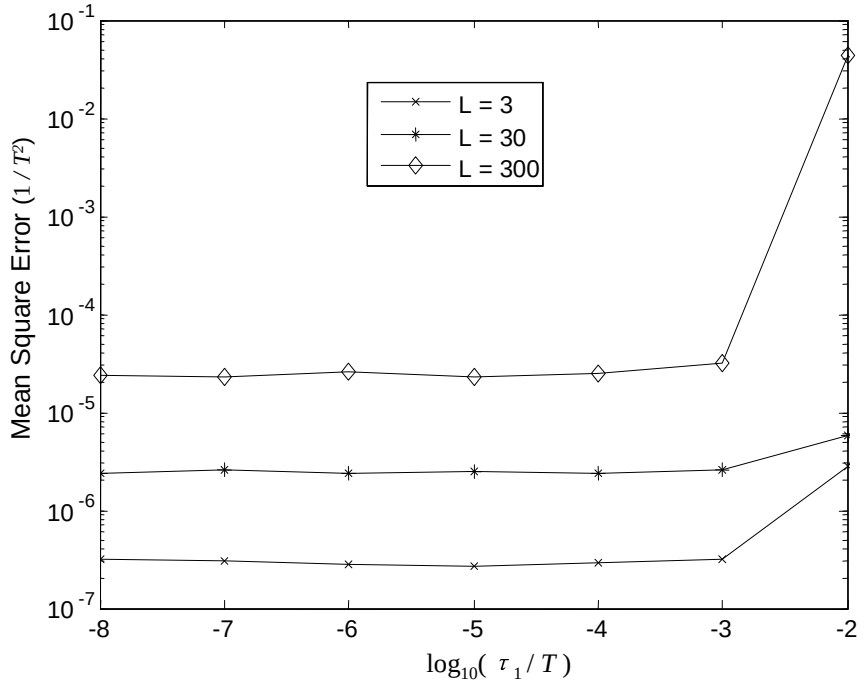


Figure 4.3: Mean square estimation error of NDA/TED in noiseless channel in the presence of random jitter, with $N = 4$ and $\sigma_t^2 = 3 \times 10^{-8}$.

complexity is traded for more accurate estimates for each symbol in a frame.

4.4 Iterative symbol timing recovery

It is shown in Section 4.3 that the time walk imposes irreducible estimation error in the feedforward NDA/TED. Intuitively, if the precision of time estimation is increased, then the overall system performance can be improved accordingly. As indicated in the system diagram of Fig. 4.1, decision feedback iterations are invoked to refine the timing estimate and improve the overall receiver performance. Soft decision is made on the log-likelihood output of the turbo decoder.

$$d_k^i = \tanh\left(L_k^i/2\right), \tag{4.19}$$

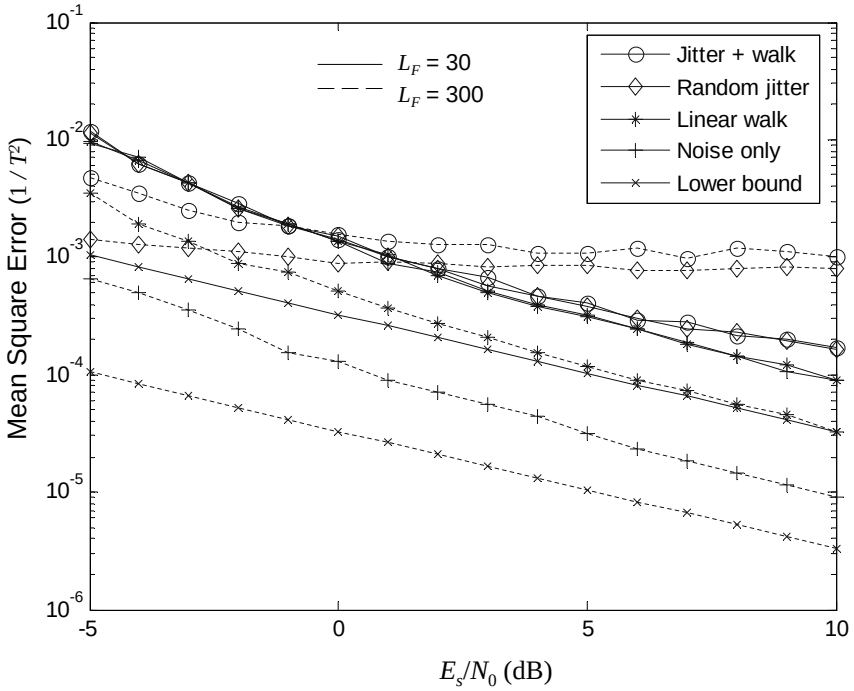


Figure 4.4: Mean square timing estimation errors of feedforward estimators for observation window sizes $L_F = 30$ and $L_F = 300$ with $N = 4$, $\tau_1 = 10^{-3}$, and $\sigma_t^2 = 10^{-6}$. BPSK modulated random data with rectangular pulse shaping is used.

where L_k^i is the log-likelihood ratio of the k-th symbol in the i-th decoding iteration. The decision-feedback TED introduced by Mueller and Müller (M&M) [5] is used. Let s_k^i denote the interpolation result and d_k^i be the intermediate result in the i-th global iteration of the k-th symbol. The error is calculated as

$$e_k^i = d_{k-1}^i s_k^i - d_k^i s_{k-1}^i. \tag{4.20}$$

A moving average is used to implement the loop filter

$$\psi_k^i = \frac{\omega}{L_B} \sum_{l=0}^{L_B-1} e_k^i. \tag{4.21}$$

The time delay estimate for each symbol is updated individually by

$$\hat{\tau}_k^{i+1} = \hat{\tau}_k^i - \psi_k^i. \tag{4.22}$$

One issue with M&M method is that it works well only when the bit error rate (BER) is less than 10^{-2} [7]. Hence it is necessary to only use reliable decision for the data, which means that the receiver must wait a few local iterations before the first decision-directed timing estimation can start.

The number of local iterations must be specified to evaluate the computational complexity of the whole system. In this research, the receiver runs a total of ten decoding iterations. If decision-directed feedback estimate is used, then the receiver runs two global iterations, with five local iterations in each global iteration.

4.5 Simulation study

The analysis and proposed techniques in previous sections are verified by Monte-Carlo simulations. A rate 1/3 turbo code is used as an example of powerful error-correction codes. It is capable of reaching BER lower than 10^{-4} at $E_b/N_0 = 1$ dB in AWGN channels. The interleaver size is 1024. The constraint length of the constituent convolutional code is 4. The turbo decoder runs a total number of 10 decoding iterations. Random data are encoded, and (root) raised-cosine pulse shaping is used with fall-off factor $\alpha = 0.5$. The received signal is corrupted by additive white Gaussian noise. The ideal sampling rate is $N = 4$, while linear walk and random jitter may exist. τ_1 and σ_t^2 are normalized about T.

When the observation windows are non-overlapped, the estimated timing delay is applied to the entire observation window, i.e., all the symbols in the sliding window are assumed to have the same time offset. This setting makes the symbols close to the borders of the sliding

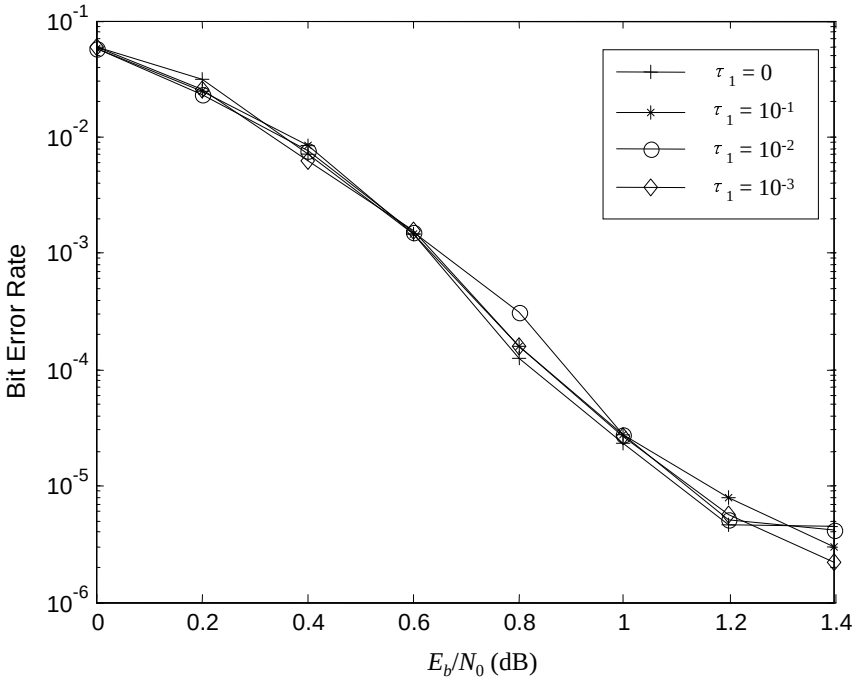


Figure 4.5: Bit error rate performance of turbo coded system with known linear walk. Timing delay is estimated by (4.9) and (4.10). Observation windows are not overlapped.

window have less accurate timing estimate. In extreme cases, this setting may cause cycle slips when L_F is large and yield burst errors. A channel interleaver is used to combat this deficiency.

When τ_1 is known to the receiver, the NDA/TED uses the exact value of N_f as shown in (4.9) and (4.10). Fig. 4.5 shows the BER curves of the turbo coded system with known fractional sampling rate when $\sigma_t^2 = 0$. As pointed out in Section 4.3.1, there is no particular difference when τ_1 changes. Although the same data and noise sequences were used in the simulation, the results are not exactly the same because the samplers were sampling the sequences at different timing. Therefore the turbo decoders have different input values and decoding results.

If τ_1 is unknown to the receiver, then the NDA/TED uses $N = 4$. In the feedforward-only case, timing estimates are generated once by the NDA/TED and the decoder iterates

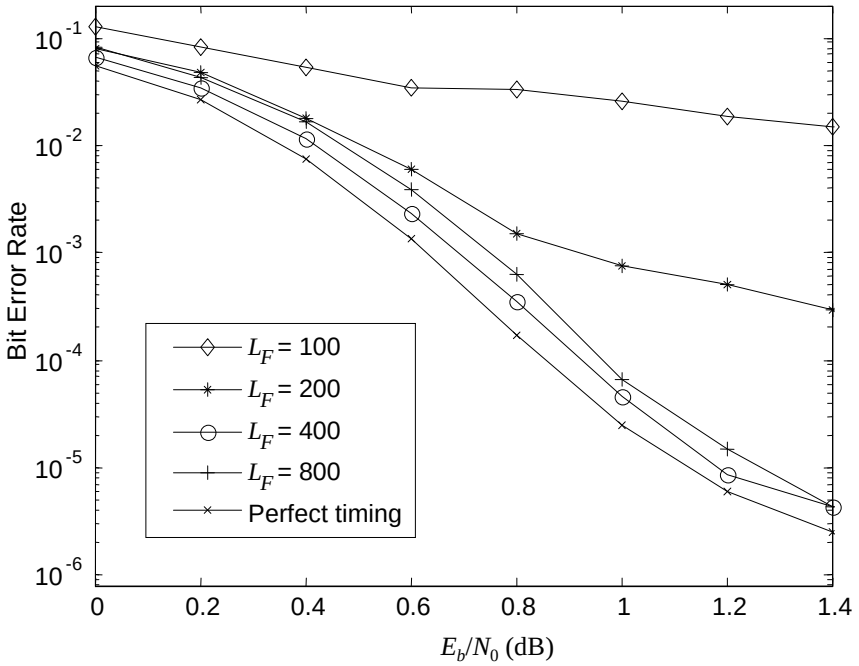


Figure 4.6: BER performance of feedforward-only turbo coded system when $\tau_1 = 10^{-4}$ and $\sigma_t^2 = 0$. τ_1 is unknown to the receiver. Observation windows are not overlapped.

10 times until the log-likelihood ratio is used for hard detection of the transmitted data. Fig. 4.6 shows the BER performance when linear walk exists and no random jitter using non-overlapped windows. The case of interest is when $\tau_1 = 10^{-4}, \sigma_t^2 = 0$, the NAD/TED uses different observation window sizes, $L_F = 100$, 200, 400, and 800. The two curves of $L_F = 100$ and 200 shows that these window sizes are too small to provide accurate timing estimate. $L_F = 400$ is superior to $L_F = 800$ because the latter is so large that cycle slips appear at the borders of the observation windows.

Fig. 4.7 shows the BER curves when $\tau_1 = 0, \sigma_t^2 = 10^{-5}$ with non-overlapped windows. The same values of L_F as in Fig. 4.6 are tested, and similar curves are obtained. Because the timing jitter is random, it is shown that when $L_F = 400$, the probability of cycle slips is lower than that when $L_F = 800$.

Fig. 4.8 shows the BER curves when $L_F = 400$ with various combinations of τ_1 and σ_t^2

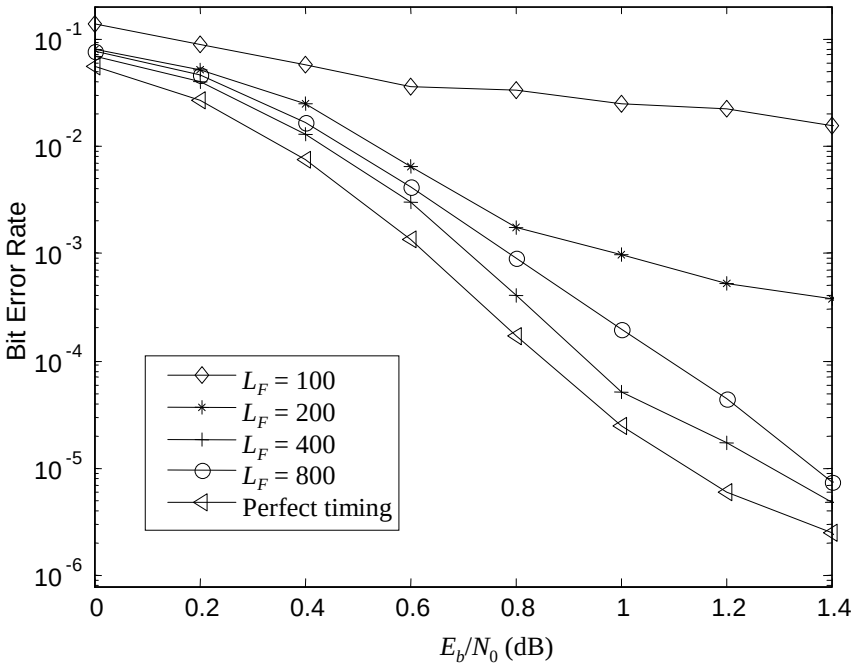


Figure 4.7: BER performance of feedforward-only turbo coded system when $\tau_1 = 0$ and $\sigma_t^2 = 10^{-5}$. τ_1 is unknown to the receiver. Observation windows are not overlapped.

using non-overlapped windows. When $\tau_1 = 10^{-3}, \sigma_t^2 = 0$ and $\tau_1 = 0, \sigma_t^2 = 10^{-4}$, the system fails to synchronize. When $\tau_1 = 10^{-4}$ and $\sigma_t^2 = 10^{-5}$, the two effects of linear walk and random jitter add up to degrade the BER performance even more severely. Error floors are observed when $\sigma_t^2 \neq 0$.

The proposed techniques to mitigate time walk are examined. Overlapped sliding windows are used to combat the cycle slips caused by linear walk. The factor γ indicates the proportion of one observation window overlapped by adjacent windows. For example, if $\gamma = 90\%$, and $L_F = 400$, then each window is shifted by $2 \times (1 - 90\%) \times 400 = 80$ symbols. The estimated time delay in this sliding window is applied to the 80 symbols in the center.

Decision-feedback is also tested to refine the time estimate. The decoder iterates five times before output decision to the DD/TED. Samples are re-interpolated by the refined timing estimate and extrinsic information from previous iterations is reused in the rest five

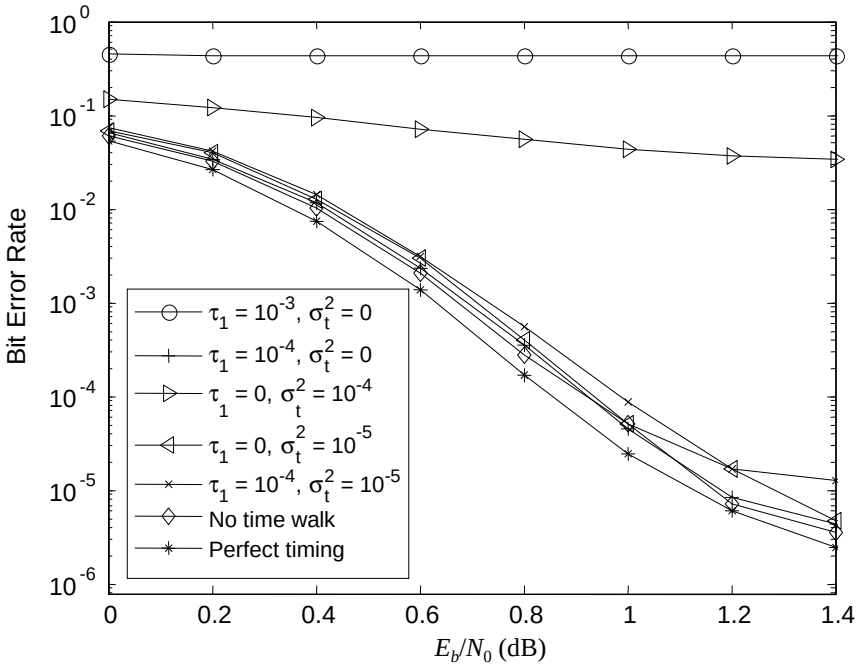


Figure 4.8: BER performance of feedforward-only turbo coded system when $L_F = 400$. Observation windows are not overlapped.

decoding iterations. $\omega = 0.5$ and $L_B = 400$ are selected so that maximal improvements on BER performance are achieved.

Fig. 4.9 demonstrates the effect of overlapped sliding window and decision feedback timing refinery when linear time walk exists. It compares the BER curves of feedforward and feedback schemes, and non-overlapped and overlapped sliding windows with $\tau_1 = 4 \times 10^{-4}$, $\sigma_t^2 = 0$, and $L_F = 400$. At BER of 10^{-3}, there is a gap of about 0.4 dB from the BER curve of using non-overlapped sliding window and feedforward-only timing estimate to the curve with no time walk ($\tau_1 = 0, \sigma_t^2 = 0$). If the sliding window are 75% overlapped, the gap decreases to about 0.2 dB at BER of 10^{-3}. When the sliding windows are non-overlapped, decision feedback timing refinery helps to recover more than 0.1 dB SNR losses at BER of 10^{-3}. However, when 75% overlapped observation windows are used, the decision feedback only slightly improves the BER performance. The overlapped sliding window and iterative processing fail to provide further improvement when $E_b/N_0 > 1$ dB and an error floor is

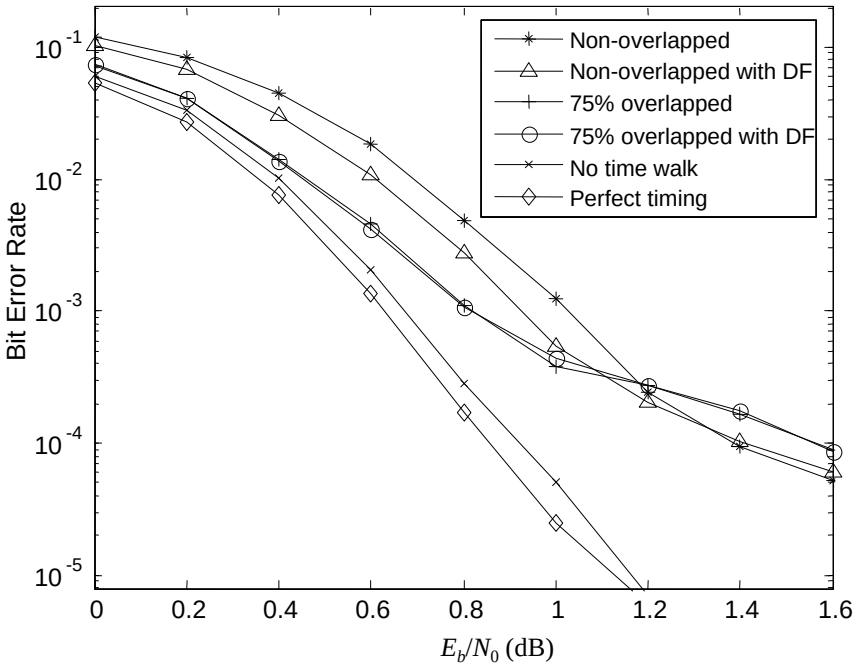


Figure 4.9: BER performance of decision-feedback iterative timing recovery when $L_F = 400$, $\tau_1 = 4 \times 10^{-4}$, and $\sigma_t^2 = 0$. The DF/TED uses $L_B = 400$ and $\omega = 0.5$.

present.

Fig. 4.10 illustrates the effect of overlapped sliding window and decision feedback timing refinery when random jitter exists. It is shown that 75% overlapped sliding window can improve the BER performance as much as 0.15 dB. When non-overlapped sliding windows are used, the BER curve is shifted towards the left by approximately 0.05 dB. When 75% overlapped windows are used, decision feedback can only slightly improve the performance. The BER curves with time walks reaches error floors when $E_b/N_0 > 0.8$ dB.

4.6 Conclusion

The effect of random walk is studied. The timing estimate error is not only related to the random walk, but also related to the size of observation window. To mitigate the negative

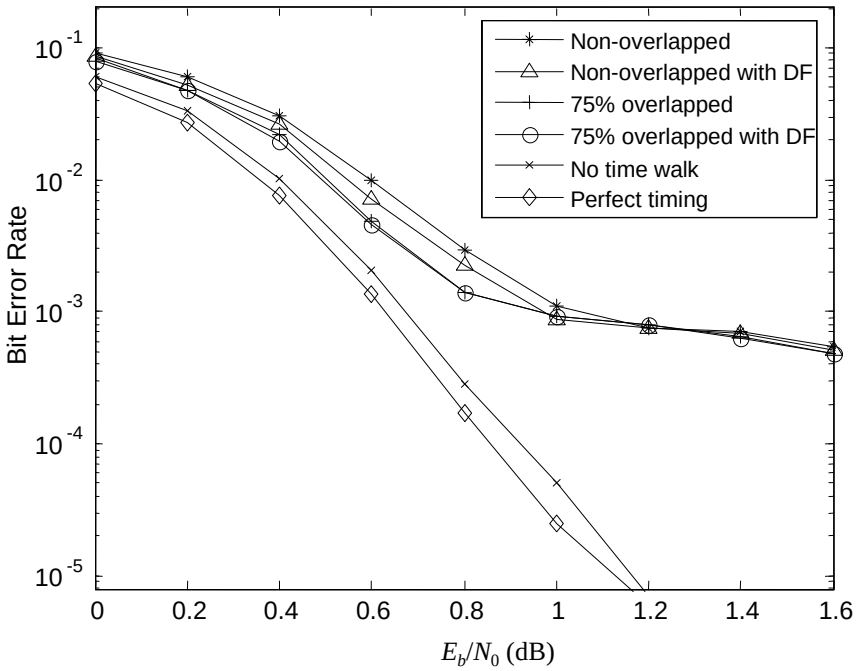


Figure 4.10: BER performance of decision-feedback iterative timing recovery when $L_F = 400$, $\tau_1 = 0$, and $\sigma_t^2 = 4 \times 10^{-5}$. The DF/TED uses $L_B = 400$ and $\omega = 0.5$.

effects of time walk, we propose the following methods to design a turbo coded system with random time walks.

1. Channel interleaver must be used to combat burst errors.

2. Appropriate observation window size must be selected to compromise estimation precision and the possibility of cycle slips.

3. Overlapped sliding windows can be used to mitigate the effect of linear walks.

4. Iterative timing refinery can be used to improve the BER performance.

Combination of the following requirements should be fulfilled to select the proper window size

1. For linear walk, $L_F \tau_1 N$ should be less than one-tenth of T.

2. For random jitter, $\mathbf{Pr}\left[\tau_2 > T/N\right] \ll 1$.

Both channel scramblers and overlapped sliding windows remarkably improve the BER performance. It is found that the decision feedback timing refinery can only contribute slight improvement to the BER performance especially when 75% overlapped observation windows are used. Error floors of the BER curves are observed when significant time walk exists. The error floor is produced by cycle slips. Further work to enable the system of detecting and correcting cycle slips would improve performance.

Chapter 5

Optimum Frame Synchronization for LDPC Codes

A MAP frame synchronization method in the sense of minimizing the probability of frame sync error for LDPC coded system is introduced. This method is based on properties of low-density parity-check and does not require the insertion of sync words or preambles. The algorithm computes the LLR of receiving even number of 1's at each check node. The sum of all LLR values, which is related to the probability of receiving a valid codeword, forms a metric on the probability of frame synchronization. This metric is a function of candidate frame starting point. The position with highest value is selected. For the considered LDPC codes, frame sync failure rates lower than 10^{-2} are achieved at E_b/N_0 less than 2 dB.

The proposed frame synchronizer is applicable to turbo codes exploiting the low-density parity-check properties of turbo codes. A turbo code is described as a special LDPC code. By using an additional scrambler, the MAP frame synchronizer also works for turbo codes. For the considered turbo codes, frame sync failure rates lower than 10^{-4} are achieved at E_b/N_0 less than 3 dB.

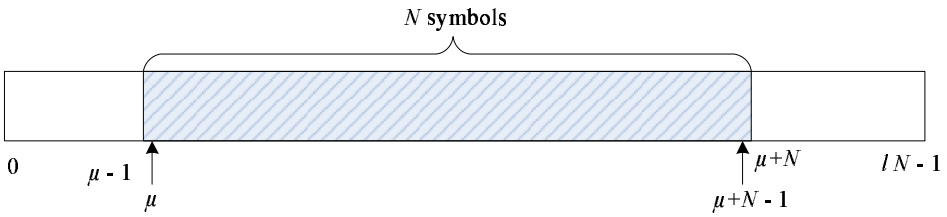


Figure 5.1: The buffer structure and estimation issues.

5.1 System model

The frame synchronization problem is presented in this section. A packet of data is LDPC encoded, transmitted, and received, corrupted by additive noise. Let $\mathbf{d} = \{d_i\}$ denote the transmitted signal, and $\mathbf{w} = \{w_i\}$ be i.i.d. additive Gaussian noise. w_i has zero mean and variance $N_0/2$, where N_0 is the one-side power spectrum density of additive noise. The received signal $\mathbf{y} = \{y_i\}$ is

$$\mathbf{y} = \mathbf{d} + \mathbf{w} \tag{5.1}$$

Assuming each symbol is sampled once with perfect symbol timing synchronization, the samples of received signal are stored in a buffer as shown in Fig. 5.1. The location μ_0 where the codeword starts is unknown. It is assumed that the codeword is completely contained in the buffered samples. This assumption is valid if a coarse frame estimator is available, for example, by using the carrier power sensor as in [24]. The buffer size is lL, where L is the codeword length and $l > 1$ is the normalized observation window size. The problem is to estimate μ_0, $0 \leq \mu_0 \leq lL - L$, from the whole frame of samples $\mathbf{y} = \{y_i\}$, $0 \leq i < lL$. If the estimate $\mu = \mu_0$, then frame synchronization is achieved. Otherwise, there is a failure.

The frame synchronization problem is a detection problem. The problem is to examine $\mathbf{y}$ against two hypotheses. The null hypothesis $\mathcal{H}_0$ means that there exist cycle slips of a few symbols. The alternative hypothesis $\mathcal{H}_1$ is that frame synchronization is achieved.

The maximum a posteriori probability (MAP) frame synchronizer in the sense of maximizing frame sync acquisition rate maximizes the following probability

$$\Pr\left[\mu\,|\mathbf{y}\right],\ \mu\in[0,lL-L] \tag{5.2}$$

which means the probability of μ when receiving the sample frame $\mathbf{y}$. It denotes the *a posteriori* probability (APP).

The following components of the samples in Fig. 5.1 are taken into account. At position $0\le i<\mu$ and $\mu+L\le i<lL$, no data is sent, therefore $d_i=0$, and it is referred to as a blank. Assuming that BPSK modulation is used, then $d_i=\pm1$ for $\mu\le i<\mu+L$. $\mathcal{C}$ is the set of all valid codewords in GF(2), $\mathcal{C}=\{\mathbf{c}:\mathbf{c}\mathbf{H}^T=\mathbf{0}\}$. Let $\tilde{\mathcal{C}}$ denotes the modulated version of $\mathcal{C}$. If $\{d_i,\cdots,d_{i+L-1}\}$ is modulated from a valid codeword $\mathbf{c}$, then $\{d_i,\cdots,d_{i+L-1}\}$ is also a valid codeword in the sense that $\{d_i,\cdots,d_{i+L-1}\}\in\tilde{\mathcal{C}}$.

The maximizing problem is now

$$\begin{aligned}
&\Pr\left[\mu\,|\mathbf{y}\right]\\
=\ &\Pr\left[\{d_0,\cdots,d_{\bar{\mu}-1}\}=\mathbf{0}\,|\mathbf{y}\right]\cdot\Pr\left[\{d_{\bar{\mu}-1},\cdots,d_{\bar{\mu}+L-1}\}\in\tilde{\mathbf{C}}\,|\mathbf{y}\right]\\
\cdot\ &\Pr\left[\{d_{\mu+L},\cdots,d_{lL}\}=\mathbf{0}\,|\mathbf{y}\right]\\
=\ &\left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{lL-L}\prod_{i=0}^{\mu-1}\exp\left(-\frac{y_i^2}{2\sigma^2}\right)\prod_{i=\mu+L}^{lL-1}\exp\left(-\frac{y_i^2}{2\sigma^2}\right)\\
\cdot\ &\Pr\left[\{d_\mu,\cdots,d_{\bar{\mu}+L-1}\}\in\tilde{\mathbf{C}}\,|\mathbf{y}\right]
\end{aligned} \tag{5.3}$$

The two products in (5.3) account for the blanks in head and tail, where only noise presents. The last probability term in (5.3) requires a decoder because it examines the argument if $\{d_\mu,\cdots,d_{\mu+L-1}\}$ is a valid codeword, where inference is drawn on the observation of $\{y_\mu,\cdots,y_{\mu+L-1}\}$. The frame synchronizer diagram is illustrated in Fig. 5.2. Implementation

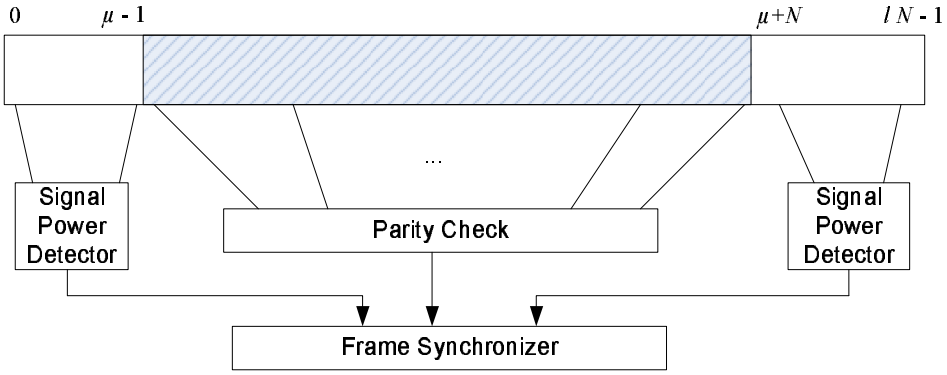


Figure 5.2: Diagram of frame synchronizer.

issues will be discussed in the following section.

5.2 LDPC codes and gaussian approximation

LDPC codes are based on low-density parity-check matrices $\mathbf{H}$. The dimension of $\mathbf{H}$ is $M \times L$. M is the number of parity-check bits. $K = L - M$ is the data word length. Regular codes are considered in this context where the row weight W_r and column weight W_c are constants for all rows and columns respectively. The proposed method is also applicable to irregular codes. Assuming BPSK modulation and additive white Gaussian noise (AWGN) channel, then the received signal $\mathbf{y}$ ignoring blanks is a sequence of antipodal signals corrupted by additive noise.

Let R_i denotes the set of indices that $R_i = \{j : h_{ij} = 1\}$ and $\mathbf{c}_i = \{c_j : j \in R_i\}$ where c_j is the jth bit in $\mathbf{c}$. The parity check equation is satisfied when there are an even number of ones in $\mathbf{c}_i$. This probability can be computed by the decoder using

$$q_i = \Pr\left[\text{Even number of 1's in } \mathbf{c}_i\right] = \frac{1}{2} + \frac{1}{2} \prod_{j \in R_i} (1 - 2p_j). \tag{5.4}$$

where

$$p_j = \mathbf{Pr}\left[c_j = 1 \,|y_j\right] \tag{5.5}$$

In the log-domain, the log-likelihood ratio (LLR) is

$$Q_i = \log\left(\frac{q_i}{1-q_i}\right) = \prod_{j\in R_i} \alpha_j \cdot \phi\left[\sum_{j\in R_i} \phi\left(\beta_j\right)\right] \tag{5.6}$$

where

$$\alpha_j \quad = \text{sign}\left[\log\left(\frac{p_j}{1-p_j}\right)\right], \quad \beta_j = \left|\log\left(\frac{p_j}{1-p_j}\right)\right|.$$

and

$$\phi\left(x\right) \quad \triangleq -\log\tanh\left(\frac{1}{2}x\right) = \log\frac{e^x+1}{e^x-1}$$

Due to the low-density property of the parity-check matrix, there are only a few entries involved in each parity-check equation.

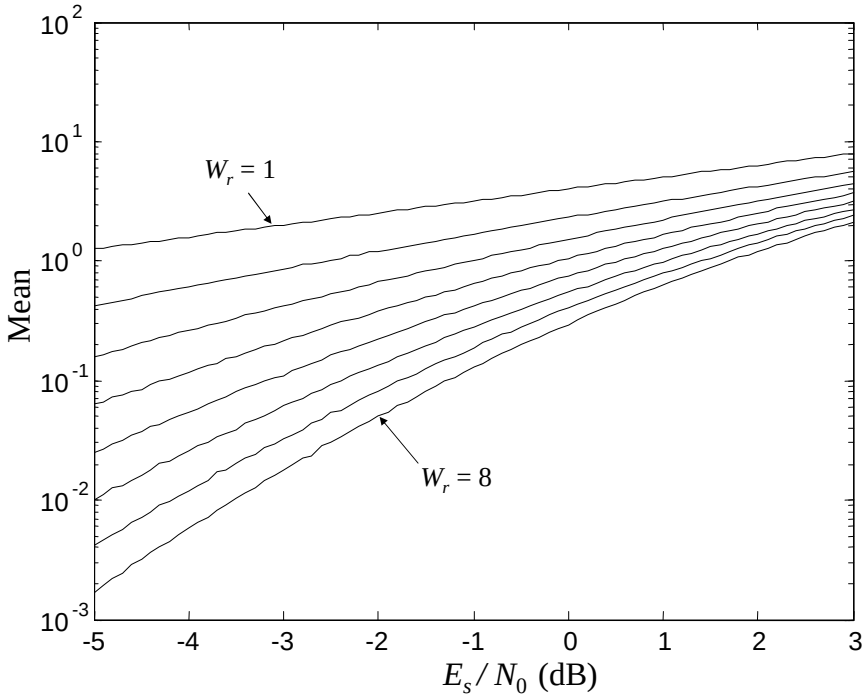
Provided $\mathbf{c}$ is a valid codeword, then $\{Q_i\}$ is a set of identically distributed random variables, which can be approximated as Gaussian distributed random variables, denoted as $\mathcal{N}\left(m_c, 2m_c\right)$ [25]. The statistics about Q_i are derived from simulations as shown in Fig. 5.3. The mean and the variance of Q_i with variant row weights are recorded. It is verified that the variance is exactly twice as much as the mean. Both the mean and the variance increase when SNR grows. Even though that the above Gaussian approximation is not accurate for large W_r, we can use central limit theorem to approximate the distribution of $\sum Q_i$ as $\mathcal{N}\left(Mm_c, \kappa Mm_c\right)$. This approximation is valid because M is large. The coefficient κ is related to W_c because for a regular LDPC code, each variable node is connected to W_c adjacent

check nodes. Therefore the Q_i's related to these W_c check nodes are not independent. $\kappa_0 = (2W_c - 1)$ is found to be a good approximation of κ.

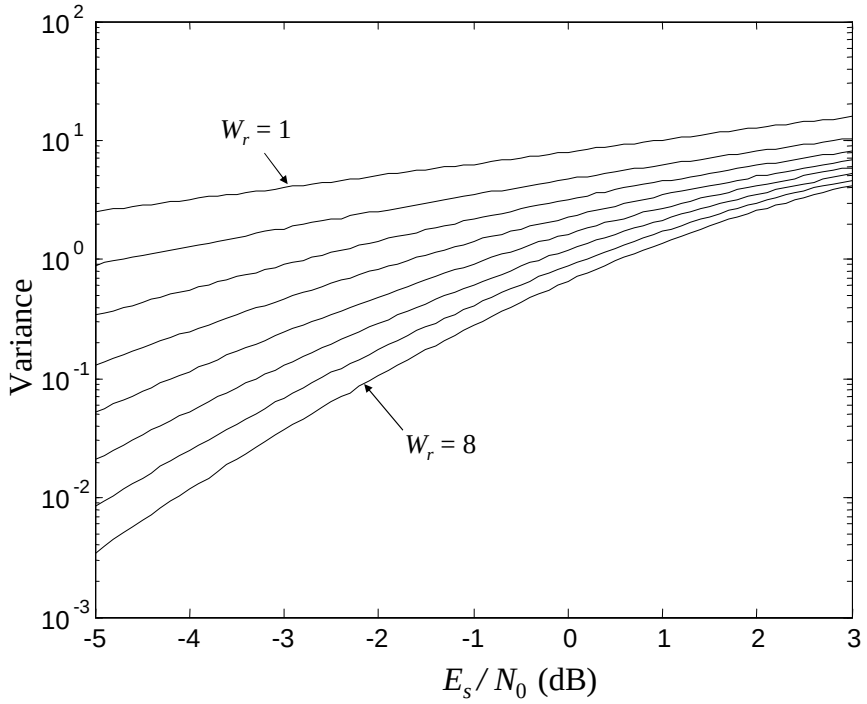
The statistic $\nu(\mu) = \sum Q_i$ is of great importance in the following discussion. It is denoted as a function of μ because given samples of $\mathbf{y}$, $\nu(\mu)$ changes along with μ. Under the assumption of correct frame synchronization, $\nu(\mu_0)$ are tested for three regular LDPC codes. These codes are generated using the code definition of MacKay [31] and generated from the codes available at [54]. Code I has a parity check matrix $\mathbf{H}$ of size (511, 1022) with $W_r = 8$. $\mathbf{H}$ of code II is (512, 1024) with $W_r = 6$. $\mathbf{H}$ of code III is (900, 1200) with $W_r = 4$. The analytical curves and simulation results about the mean and variance of $\nu(\mu_0)$ are presented in Fig. 5.4. The analytical mean is found by plotting Mm_c, while the analytical variance is $\kappa_0 M m_c$. The values of m_c are read from corresponding curves shown in Fig. 5.3. The analytical and simulation curves are very close.

If $\mathbf{c}$ is not a valid codeword, which happens when symbol slips exist $\mu \neq \mu_0$, then the parity check equations will yield positive or negative values. This randomness is due to $\mathbf{H}$. This leads to $|\mathbf{E}[Q_i]| = m_c$, however $\mathbf{E}[Q_i]$ is randomly positive or negative. Hence we have $\mathbf{E}[\nu(\mu)] \approx 0$, and $\nu(\mu) \propto \mathcal{N}(0, \kappa M m_c)$.

The distribution of $\nu(\mu)$ when $\mathbf{c}$ is synchronized and not synchronized are illustrated as the histogram in Fig. 5.5. $\nu(\mu)$ is apparently Gaussian distributed in the two cases. The distance between the distribution helps the proposed frame synchronization, which is described and analyzed below.

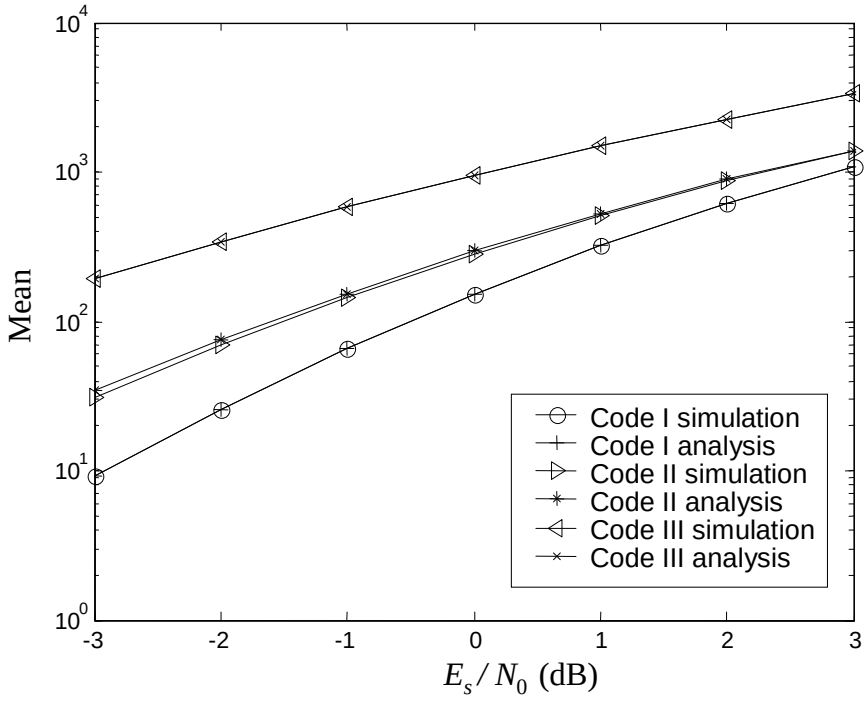


(a)

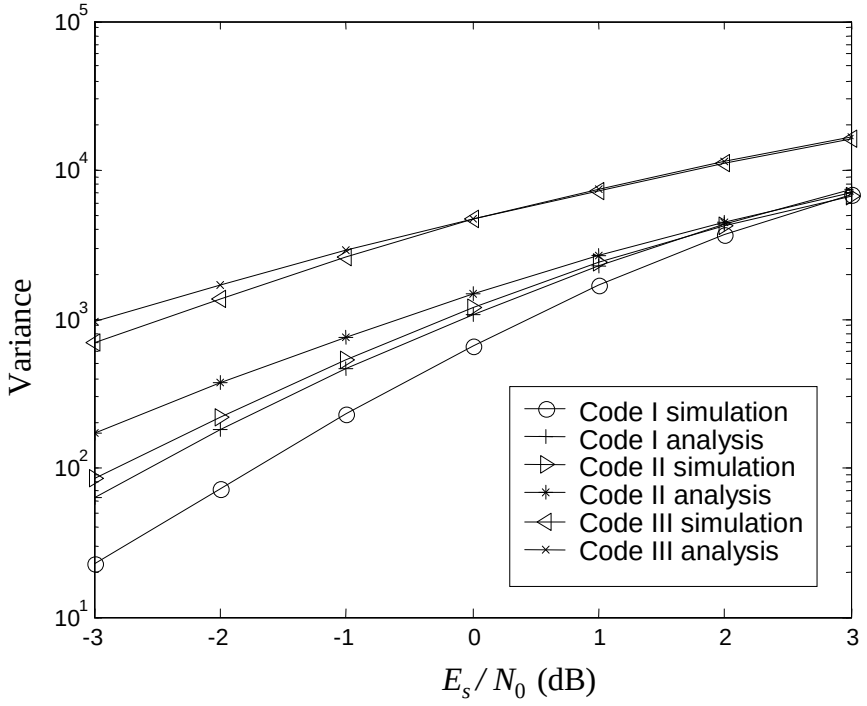


(b)

Figure 5.3: Statistics about Q_i found by simulations.



(a)



(b)

Figure 5.4: Statistics about $\nu(\mu_0)$ found by simulations for code I, II, and III. $\mathbf{H}$ of code I has the dimension of 511×1022 and $W_r = 8$. $\mathbf{H}$ of code II has the dimension of 512×1024 and $W_r = 6$. $\mathbf{H}$ of code III has the dimension of 900×1200 and $W_r = 4$.

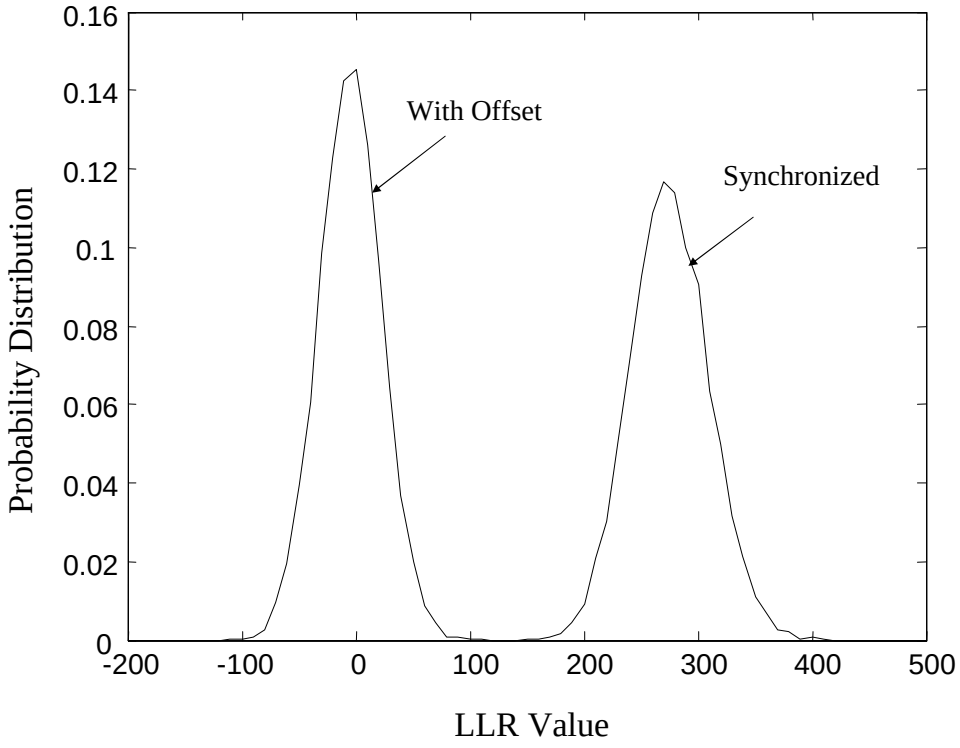


Figure 5.5: Distribution of $\nu(\mu)$ for LDPC code with code I at $E_s/N_0 = 0$ dB. $\mathbf{H}$ of code I has the dimension of 511×1022 and $W_r = 8$.

5.3 Frame synchronization implementation

5.3.1 Optimum frame synchronizer

Using Gaussian approximation presented in Section 5.2, the distribution of $\nu(\mu)$ under $\mathcal{H}_0$ is

$$f_e(x) = \frac{1}{\sqrt{2\pi\kappa M m_c}} \exp\left(-\frac{[\nu(\mu) - M m_c]^2}{2\kappa M m_c}\right). \tag{5.7}$$

The likelihood function is

$$\begin{aligned} L(\mu) &= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^{lL-L} \prod_{i=0}^{\mu-1} \exp\left(-\frac{y_i^2}{2\sigma^2}\right) \prod_{i=\mu+L}^{lL-1} \exp\left(-\frac{y_i^2}{2\sigma^2}\right) \\ &\cdot \frac{1}{\sqrt{2\pi\kappa M m_c}} \exp\left(-\frac{[\nu(\mu) - M m_c]^2}{2\kappa M m_c}\right) \end{aligned} \tag{5.8}$$

After elimination of unrelated terms, the logarithm of the above likelihood function, or the log-likelihood function, is

$$\log[L(\mu)] = -\sum_{i=0}^{\mu-1} \frac{y_i^2}{2\sigma^2} - \sum_{i=\mu+L}^{lL-1} \frac{y_i^2}{2\sigma^2} - \frac{[\nu(\mu) - M m_c]^2}{2\kappa M m_c}. \tag{5.9}$$

The computation of (5.9) involves two recursive parts and one decoding part. The last term can only be calculated using decoder. The optimum estimate of μ in MAP sense is

$$\bar{\mu} = \arg\max_{\mu} \{\log[L(\mu)]\} \tag{5.10}$$

5.3.2 High-SNR approximation

When SNR is high,

$$\Pr\left[\{d_0, \cdots, d_{\bar{\mu}-1}\} = \mathbf{0} \,|\mathbf{Y} = \mathbf{y}\right] \approx \Pr\left[d_{\bar{\mu}-1} = 0 \,|\mathbf{Y} = \mathbf{y}\right] \tag{5.11}$$

Likewise, we have

$$\Pr\left[\{d_{\bar{\mu}+L}, \cdots, d_{lL-1}\} = \mathbf{0} \,|\mathbf{Y} = \mathbf{y}\right] \approx \Pr\left[d_{\bar{\mu}+L} = 0 \,|\mathbf{Y} = \mathbf{y}\right] \tag{5.12}$$

Therefore we modify the log-likelihood function to high-SNR approximation as

$$L_{high}\left(\mu\right) = -\frac{y_{\mu-1}^2}{2\sigma^2} - \frac{y_{\mu+L}^2}{2\sigma^2} - \frac{\left[\nu\left(\mu\right) - Mm_c\right]^2}{2\kappa M m_c}. \tag{5.13}$$

5.3.3 Low-SNR approximation

When SNR is low, the first two terms in (5.9) becomes insignificant because signals are "buried" in noises. Therefore we can use the low-SNR approximation

$$L_{low}^1\left(\mu\right) = -\frac{\left[\nu\left(\mu\right) - Mm_c\right]^2}{2\kappa M m_c}. \tag{5.14}$$

Furthermore, if the occurance of the events of having a large $\nu(\mu)$ value that is greater than Mm_c when $\mu \neq \mu_0$ is scarce, i.e. $\Pr\left\{\mu \,|\nu\left(\mu\right) > Mm_c, \mu \neq \mu_0\right\} \approx 0$, then the following likelihood function is viable,

$$L_{low}\left(\mu\right) = \nu\left(\mu\right). \tag{5.15}$$

5.3.4 Frame sync failure rate

The frame synchronization failure rate is evaluated

$$\Pr\left[\text{Failure}\right] = \Pr\left[L\left(\mu'\right) > L\left(\mu_0\right) | \mu = \mu_0\right]. \tag{5.16}$$

Without loss of generality, we consider the situation where $\mu' < \mu_0$.

$$\begin{aligned} & \log\left[L\left(\mu'\right)\right] - \log\left[L\left(\mu_0\right)\right] \\ = & \sum_{i=\mu'}^{\mu_0-1} \frac{y_i^2}{2\sigma^2} - \sum_{i=\mu'+L}^{\mu_0+L-1} \frac{y_i^2}{2\sigma^2} - \frac{\left[\nu\left(\mu'\right) - Mm_c\right]^2}{2\kappa M m_c} + \frac{\left[\nu\left(\mu_0\right) - Mm_c\right]^2}{2\kappa M m_c} \end{aligned} \tag{5.17}$$

The first and second summations have $(\mu_0 - \mu')$ terms of central χ^2-distributed random variables respectively. The third term is a non-central χ^2-distributed random variable. The forth term is a central χ^2-distributed random variable.

Ignoring the first two terms in (5.17), the following probability is evaluated

$$\Pr\left\{\frac{\left[\nu\left(\mu_0\right) - Mm_c\right]^2}{2\kappa M m_c} > \frac{\left[\nu\left(\mu'\right) - Mm_c\right]^2}{2\kappa M m_c}\right\}, \tag{5.18}$$

which is reduced to

$$\Pr\left\{\left[\nu\left(\mu_0\right) + \nu\left(\mu'\right) - \kappa M m_c\right]\left[\nu\left(\mu_0\right) - \nu\left(\mu'\right)\right] > 0\right\}. \tag{5.19}$$

The probability presented in (5.19) is an approximation of (5.16) in low SNR, and it becomes an upper bound of (5.16) in high SNR. Note that $\nu\left(\mu_0\right)$ is $\mathcal{N}\left(Mm_c, \kappa M m_c\right)$, and $\nu\left(\mu'\right)$ is

$\mathcal{N}(0, \kappa M m_c)$.

$$\begin{aligned} &\Pr[\text{Failure}] \\ &= \int_0^\infty \frac{2}{\sqrt{2\pi\kappa M m_c}} \exp\left(-\frac{x^2}{2\kappa M m_c}\right) \left[\Psi\left(\frac{M m_c + x}{\sqrt{2 M m_c}}\right) - \Psi\left(\frac{M m_c - x}{\sqrt{2 M m_c}}\right)\right] \mathrm{d}x \end{aligned} \quad (5.20)$$

where $\Psi(x)$ is the cumulative distribution function of a standard normal distribution

$$\Psi(x) = \int_{-\infty}^{x} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t. \quad (5.21)$$

5.4 Parity-check characteristics of turbo codes

In order to apply the frame synchronizer described above to turbo codes, the code structure of turbo codes is defined in this section. While most work to date has viewed turbo codes from the literal perspective of being parallel concatenated recursive systematic convolutional (RSC) codes, Engdahl and Zigangirov provide an alternative way to view turbo codes as low density parity check (LDPC) codes [55]. The connection is established by the structure of convolutional codes. As early as 1973, Forney [56] suggested to transform truncated convolutional codes into linear block codes. Unlike usual LDPC codes that are defined on random sparse parity-check matrices, the linear block codes derived from turbo codes are highly structural, and in particular, they are quasi-cyclic. By "quasi-cyclic" we mean that the pattern in the parity-check matrix is repeated in the rows though the shift may be greater than one symbol.

5.4.1 Turbo encoder

Fig. 5.6 presents a diagram of a typical turbo encoder. A turbo encoder has two identical constituent RSC encoders. Encoder I uses $\mathbf{x}$ as its systematic input, while Encoder II uses

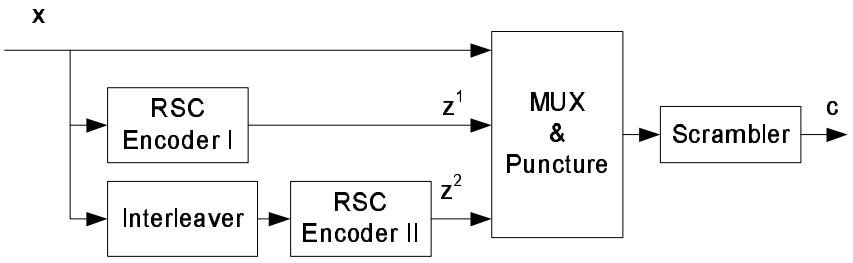


Figure 5.6: Diagram of a turbo encoder.

an interleaved version of $\mathbf{x}$ as input. The parity outputs $\mathbf{z^1}$ and $\mathbf{z^2}$, together with $\mathbf{x}$ enter a multiplexer so that the bits are assembled into a codeword. In the multiplexer, some bits in $\mathbf{z^1}$ and $\mathbf{z^2}$ are punctured in order to increase the code rate. A scrambler, also called a channel interleaver, permutates the codeword so that sequential symbols are interleaved. The permutation helps to combat burst errors which turbo codes are not good at dealing with. It also enables the frame synchronization technique proposed in this chapter. All these components in the encoder determine the parity-check matrix $\mathbf{H}$. The code structure is analyzed as following.

5.4.2 Constituent RSC codes

We start with non-systematic convolutional (NSC) codes. If an NSC code has the generating matrix as $\mathbf{G}(D) = [g_1(D) \quad g_2(D)]$, then its dual code is defined by the matrix $\mathbf{H}(D) = [g_2(D) \quad g_1(D)]$. For example, let $\mathbf{G}(D) = [1 + D + D^2 \quad 1 + D^2]$, the correspond-

ing $\mathbf{H}(D) = [1 + D^2 \quad 1 + D + D^2]$, and the matrix in numerical form is [57]

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 1 & \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & \\ & & & & & & \cdots & & & & & & \end{bmatrix}. \tag{5.22}$$

Most entries in $\mathbf{H}$ are "0". This sparseness of $\mathbf{H}$ makes it resemble the parity check matrix of an LDPC code except that it is cyclic. Each NSC code has its equivalent recursive systematic convolutional (RSC) code. In the field of GF (2), if the NSC code is $\mathbf{G}(D) = [g_1(D) \quad g_2(D)]$, then the generating matrix of the RSC code is $\mathbf{G}_1(D) = [1 \quad g_2(D)/g_1(D)]$. Because the code space remains the same, the $\mathbf{H}$ matrix of the RSC code is the same as that of the NSC code.

5.4.3 Puncturing

Puncturing is frequently used to increase the coding rate. The puncturer deletes some of the parity bits. Those columns in $\mathbf{H}$ corresponding to these bits should not be included in any parity check equation. Puncturing reduces the number of parity-check equations, as well as the number of rows in the parity-check matrix. For example, if the puncturing rule is to delete every other parity bit, and the original codeword is $\mathbf{c} = [x_0\, p_0\, x_1\, p_1\, x_2\, p_2\, x_3\, p_3\, x_4\, p_4]$, then the puncturing result is $\mathbf{c}' = [x_0\, p_0\, x_1\, x_2\, p_2\, x_3\, x_4\, p_4]$. Using $\mathbf{H}$ in (5.22) as an example,

the new parity-check matrix is

$$\mathbf{H}' = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 & \cdots \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & \\ & & & & & & \cdots & & & & & & \end{bmatrix}. \tag{5.23}$$

The number of rows in $\mathbf{H}'$ reduces to one half of the original $\mathbf{H}$. The density of the parity-check matrix is also increased after puncturing, especially the number of 1's in every row.

5.4.4 Permutation and interleaving

The parity-check matrix $\mathbf{H}$, as shown in (5.22), is cyclic. Therefore any shift of a valid codeword still satisfies all parity-check equations. This is undesirable for frame synchronization because the synchronizer needs to distinguish the correct frame starting point from other positions. The interleaver and the scrambler permutate the bit sequence so that the resulting codeword is no longer cyclic. At the receiver, the bit sequence is rearranged to recover its original order.

Let $\mathbf{c}_0$ be the original codeword before the scrambler, and $\mathbf{H}_0$ be the corresponding parity-check matrix. The permutation is an elementary operation

$$\mathbf{c} = \mathbf{c}_o\mathbf{P} \tag{5.24}$$

The new parity-check matrix is then $\mathbf{H} = \mathbf{P}^{-1}\mathbf{H}_0$.

5.5 Simulation results

Simulation results are reported to verify the feasibility of proposed frame synchronization technique and represent the performance in terms of sync-failure rate.

The three codes introduced in Section 5.2 are used. $\mathbf{H}$ of code I has the dimension of 511×1022 and $W_r = 8$. $\mathbf{H}$ of code II has the dimension of 512×1024 and $W_r = 6$. $\mathbf{H}$ of code III has the dimension of 900×1200 and $W_r = 4$. BPSK modulation is used with AWGN channel. The observing window size is 30 bits longer than the codeword length.

Fig. 5.7 illustrates the synchronizers' performance in terms of sync failure rate. For all three codes, the proposed synchronizer achieves frame sync failure rates lower than 10^{-2} at $E_b/N_0 < 2$ dB. Analytical results are provided using (5.20), where $\kappa_0 = (2W_c - 1)$ is used in the places of κ. The analytical curves are close to simulation results at low SNR and serve as upper bounds at high SNR as stated in Section 5.3.4.

Code I and code II has the same code rate and similar dimension of $\mathbf{H}$. However, code I has greater W_c and W_r. Therefore, the synchronizer works better for code II because the distributions of synchronized $\nu(\mu_0)$ and offset $\nu(\mu_0)$ fall farther apart than those of code I.

Although code III has the smallest W_r and a very large M, the performance is inferior to code II because its code rate is only 1/4. Therefore the values of E_b/N_0 are translated into much lower E_s/N_0.

It is necessary for the frame synchronizers to produce a frame sync failure rate that is acceptable for the decoders. Fig. 5.8 shows frame error rates of LDPC decoders with perfect frame synchronization and with ML frame synchronizers for code I, II, and III. For all the codes, the frame error rates are close to the frame error rates with perfect frame synchronization. Therefore the frame sync failure rate only has negligible effect on the overall frame error performance.

Packet transmission of turbo codes in AWGN channels is simulated with BPSK modula-

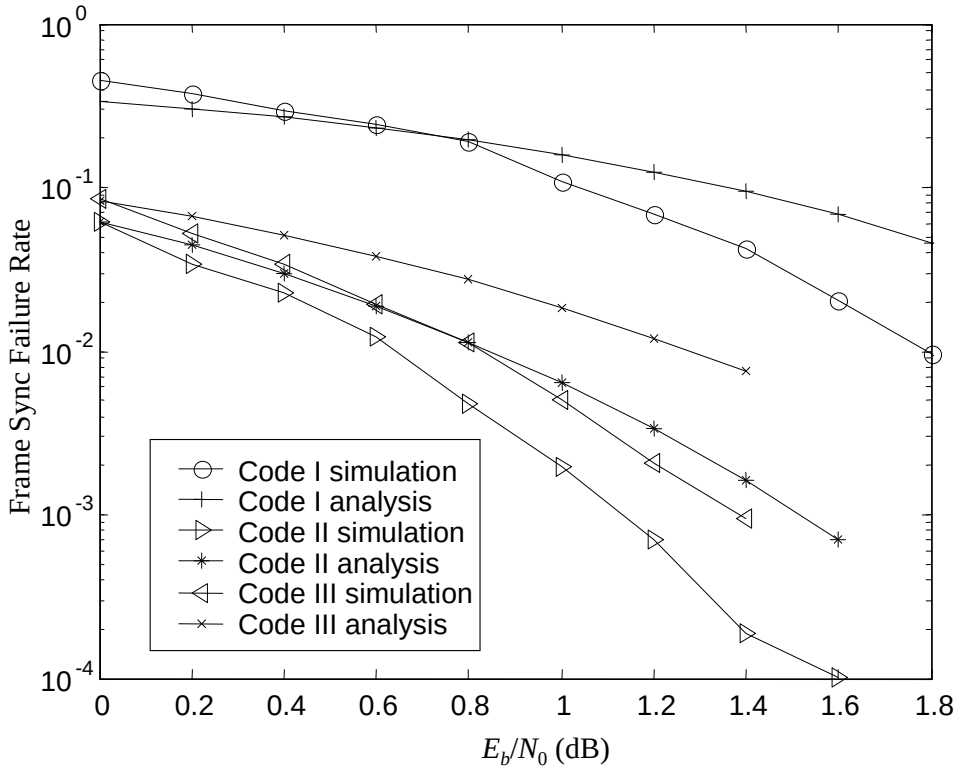


Figure 5.7: Frame synchronization failure rates. **H** of code I has the dimension of 511×1022 and $W_r = 8$. **H** of code II has the dimension of 512×1024 and $W_r = 6$. **H** of code III has the dimension of 900×1200 and $W_r = 4$.

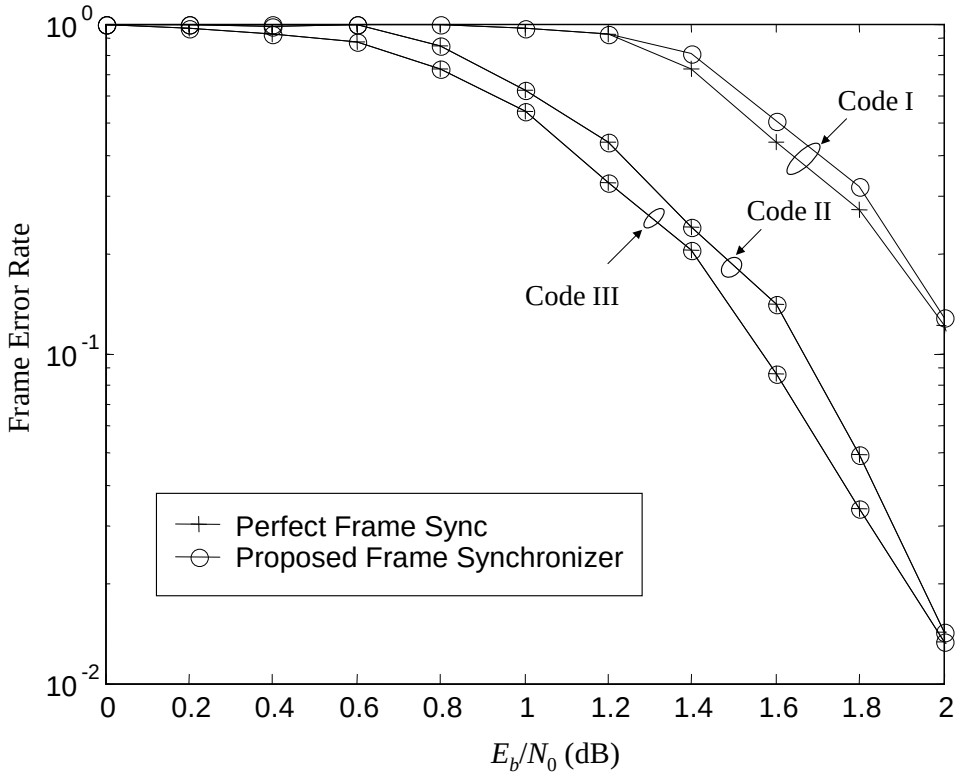


Figure 5.8: Frame error rate with perfect frame synchronization and frame sync failure rate with ML synchronizer.

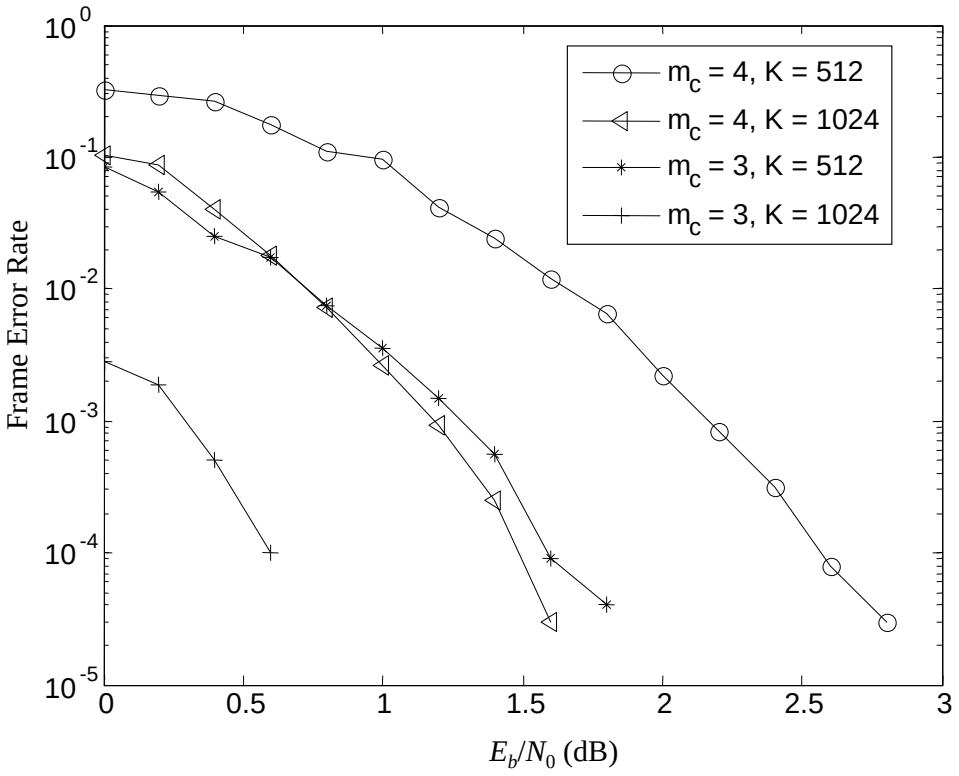


Figure 5.9: Frame sync failure rate of turbo codes with code rate 1/3. Constraint lengths of constituent RSC codes are $k_c = 3$ and $k_c = 4$ respectively. Random interleaver and scramblers are used.

tion. Simulation results of rate 1/3 turbo codes are plotted in Fig. 5.9. Two families of turbo codes are tested with random interleaver and scrambler. One family of codes have constraint length $m_c = 3$. The constraint length of the other family is $k_c = 4$. The interleaver sizes considered are $K = 512$ and $K = 1024$ respectively. $\kappa = (2W_c - 1)$ is used, where W_c is the row weight of $\mathbf{H}$. A sync failure is counted when the decision made by the frame synchronizer is not the same as the actual frame starting point. The sync failure rate curves show that the failure rate is related to both the interleaver size and constraint length. Generally, the failure rate grows when the density of $\mathbf{H}$ increases. Longer constraint length corresponds to higher density of $\mathbf{H}$. The interleaver size determines the number of check nodes. If the weight of rows and columns in $\mathbf{H}$ keeps the same, a greater interleaver size leads to lower density of $\mathbf{H}$. Therefore the frame sync failure rate is lower for codes with smaller constraint length and greater interleaver size. In all cases of interest, a frame sync failure rate lower than 10^{-4} is achieved when $E_b/N_0 < 2.5$ dB.

Fig. 5.10 compares the frame sync failure rate of punctured and original turbo codes that are not punctured. The constraint length is $k_c = 3$. Half parity bits are removed by the puncturer. Puncturing increases the density of $\mathbf{H}$. Hence it increases the sync failure rate as expected. Frame sync failure rates lower than 10^{-4} are achieved when $E_b/N_0 < 3$ dB.

Fig. 5.11 shows the frame error rate performance of punctured turbo codes. Every other parity bits in $\mathbf{y}^1$ and $\mathbf{y}^2$ are punctured to increase the code rates to 1/2. The interleaver size is 1024. It is shown that punctured turbo codes are weaker than the original codes. The performance of the proposed frame synchronizer is also affected by puncturing. When $k_c = 4$, the greatest gap between the curves of the system using the proposed synchronizer and the system with perfect synchronization is about 1 dB and the curves converge when $E_b/N_0 > 3.4$ dB. When $k_c = 3$, the curves of the system using the proposed synchronizer and the system with perfect synchronization overlap. Hence when $k_c = 3$, the proposed frame synchronizer has negligible effects on the performance of the overall system.

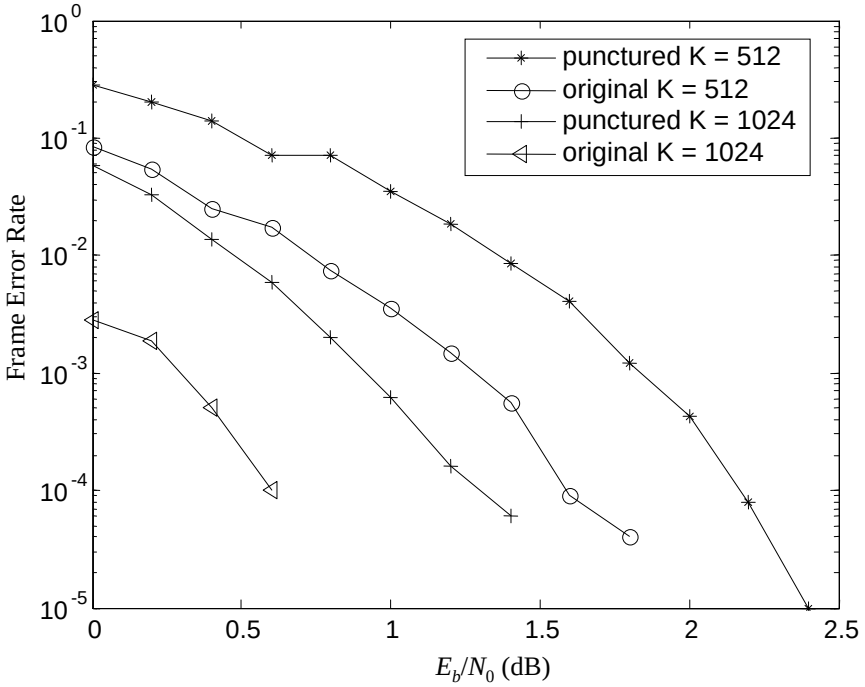


Figure 5.10: Frame sync failure rate of punctured turbo codes and original turbo codes that are not punctured. The punctured codes have code rate 1/2. Constraint lengths of constituent RSC codes are $k_c = 3$. Random interleaver and scramblers are used.

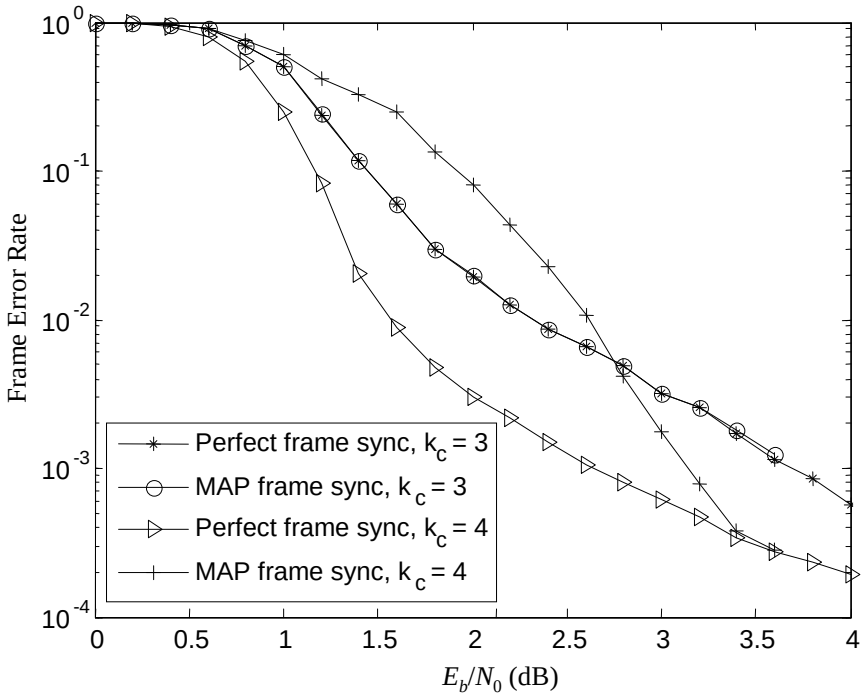


Figure 5.11: Frame error rate of turbo codes with code rate 1/2, interleaver size = 1024. Random interleavers and scramblers are used.

5.6 Conclusion

The optimum frame synchronization method based on low-density **H** is proposed for packet transmission. This method does not require insertion of sync words and performs much more reliable than conventional autocorrelation-based method. Frame synchronization is acquired before the LDPC decoder, so that no coding power is consumed by the frame synchronization module. Thereafter, the LDPC decoder can fulfill its decoding capability without coding gain loss.

Chapter 6

Conclusions: Putting It All Together

6.1 Dissertation summary

This dissertation confronts the problem of time synchronization in capacity-approaching coded systems. In this research, turbo codes are more intensively studied than LDPC codes because

- Turbo codes have clearly defined structures;
- Optimized LDPC codes are not directly available;
- Turbo codes have been used in 3G wireless communication systems.

However, all the principles are directly applicable to LDPC codes. The unknown time delay τ in (1.4) is estimated and recovered by techniques described in Chapter 3, 4, and 5.

This dissertation proposed to combine symbol timing recovery and SNR estimation. With a sampling rate as low as four samples per symbol, the bit error rate is very close to the system with ideal timing and perfect SNR knowledge. Turbo principle is applied to joint time delay and SNR estimation. With sampling rates of two or three samples per symbol, the

iterative refinement greatly improves the overall system performance at the cost of minimum additive complexity.

Random time walks may degrade the system performance severely in low SNR. Their effects are mitigated by overlapped sliding windows and iterative processing.

MAP frame synchronization in the sense to minimize frame sync failure probability is proposed for LDPC codes and turbo codes. The frame synchronizer explores the low-density parity-check attributes of LDPC codes and turbo codes. The frame synchronization utilizes the code structure and achieves reliable performance with minimum loss of coding gain.

6.2 System integration

A receiver diagram with complete timing synchronization is depicted in Fig. 6.1. To begin receiving a codeword, the switch is placed in position 1. The receiver employs a coarse frame estimator to find a rough estimation of the frame position. The frame starting point remains unknown though it is bounded in a known range. The received signal in a observation window is sampled multiple times each symbol period. The samples are used to recover proper timing and to estimate channel SNR. A frame synchronizer finds the most probable starting point and generates the desired input to the iterative decoder. After a few decoding iterations, the switch is placed at position 2. The decision is fed back to a decision aided synchronizer to refine the timing estimation. A descrambler is needed because the frame synchronizer is bypassed.

6.3 Simulation results

Integrated turbo coded systems are simulated in AWGN channels. The turbo encoders use a constraint length $k_c = 4$ where the generating polynomial coefficient is (15, 13). Two flavors

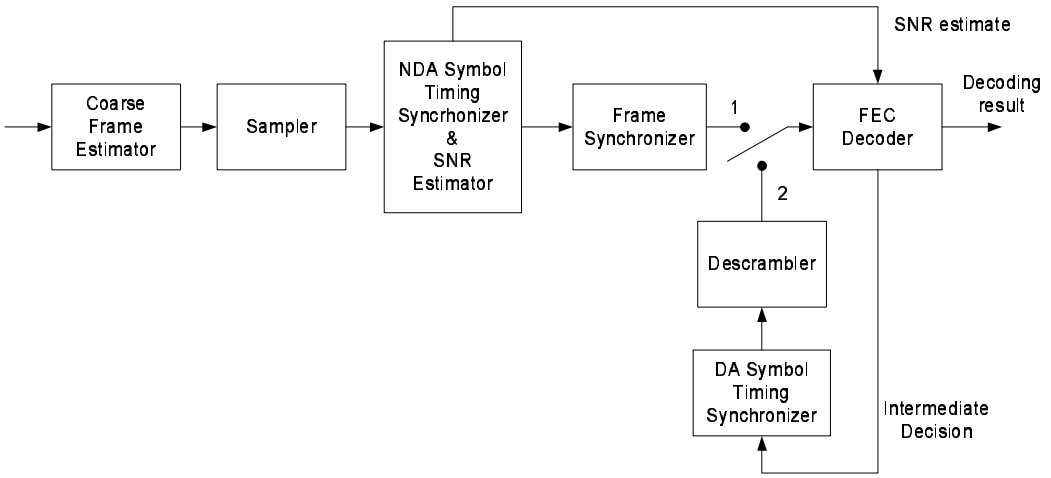


Figure 6.1: A system block diagram with integrated symbol timing synchronization and frame synchronization.

of the turbo codes are examined. One is the original with a code rate of 1/3, and the other is punctured to the code rate of 1/2. BPSK modulation is used with raised cosine roll-off pulse shaping with $\alpha = 0.5$. Random interleavers are used with interleaver size $K = 1024$. A regular or random interleaver is used.

Fig. 6.2 plots the BER curves of the rate 1/3 codes. In the legend of Fig. 6.2 and the figures hereafter, the following definitions are used.

- "Perfect" means perfect symbol timing and frame synchronization unless specifically stated.
- "Integrated" means that the system estimates both symbol timing and frame starting point.
- "Regular" means that a regular scrambler is used.
- "Random" means that a random scrambler is used.

It is shown in Fig. 6.2 that the system with perfect symbol and frame synchronization and a random scrambler has the best BER performance. The system that estimates the symbol

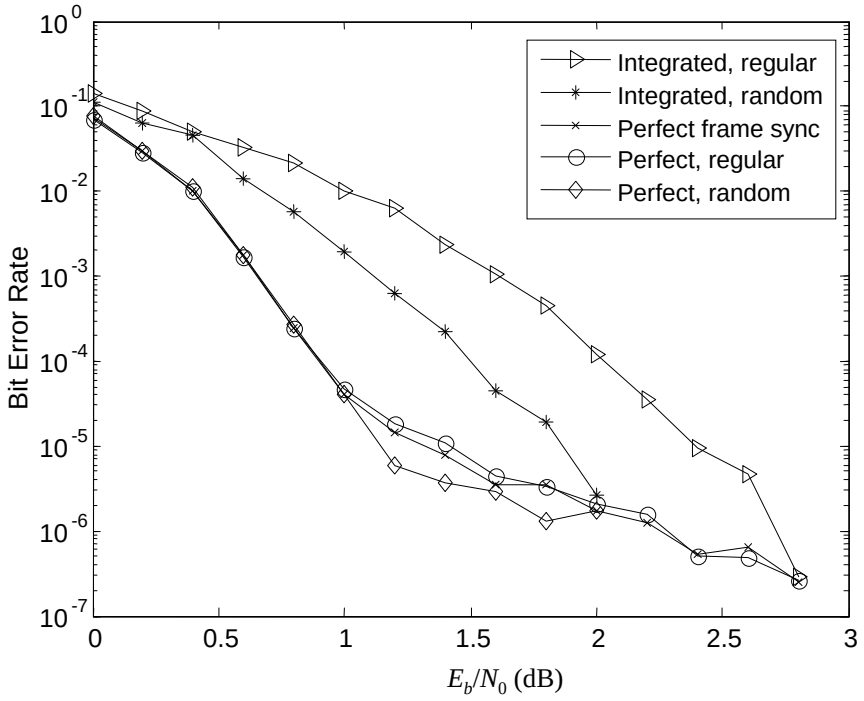


Figure 6.2: Bit error rate of turbo codes with code rate 1/3, interleaver size = 1024. Random interleavers are used. A regular or random scrambler is used. BPSK modulation with raised-cosine roll-off pulse shaping is used with $\alpha = 0.5$.

timing with perfect frame synchronization has a BER curve close to those of the systems with perfect symbol timing and frame synchronization. When perfect frame synchronization is not available, the integrated systems tend to have degraded BER performance at very low SNR because if a codeword is misaligned then the whole codeword is completely lost and cannot be recovered by any means. The BER curve of the system with a regular scrambler is about 1 dB away from the BER curve of the system with perfect synchronization at the BER of 10^{-4}. This curve converges to the one of perfect synchronization at $E_b/N_0 > 2.5$ dB. A random scrambler helps to recover the BER performance by about 0.5 dB. The BER curve of the system with a random scrambler converges to the BER curve of perfect synchronization at $E_b/N_0 > 2$ dB.

Fig. 6.3 depicts the FER curves of the turbo coded systems with a code rate 1/3. The

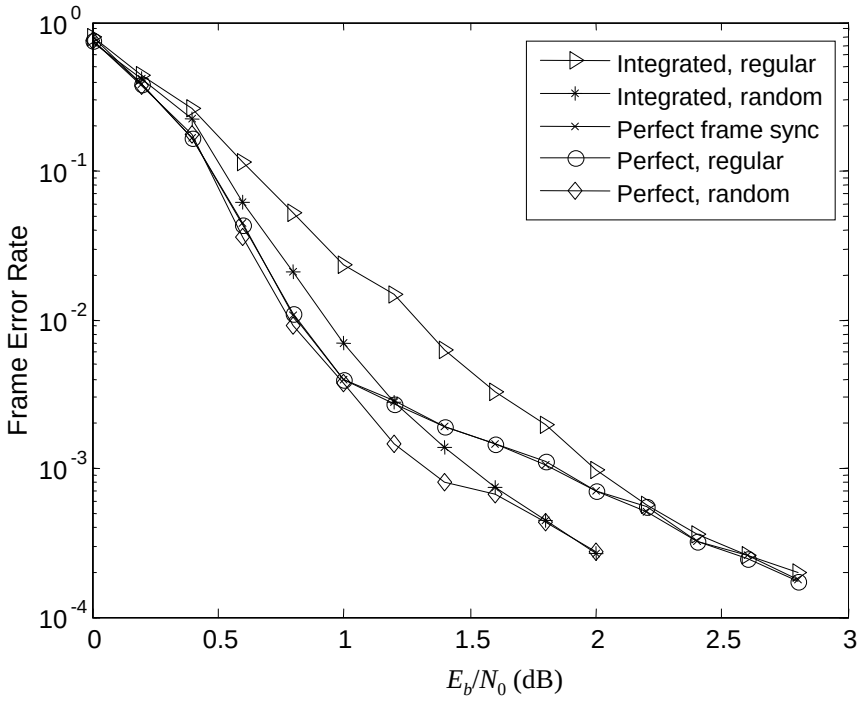


Figure 6.3: Frame error rate of turbo codes with code rate 1/3, interleaver size = 1024. Random interleavers are used. A regular or random scrambler is used. BPSK modulation with raised-cosine roll-off pulse shaping is used with $\alpha = 0.5$.

MAP frame synchronizer has performance close to the perfect synchronization. With a regular scrambler, the integrated system has FER performance that is about 0.5 dB away from the system with perfect synchronization. When a random scrambler is used, the gap is less than 0.1 dB. Fig. 6.3 shows that the systems with random scramblers are superior to systems with regular scramblers. This phenomenon happens because errors caused by ISI are mitigated.

Fig. 6.4 and Fig. 6.5 show the BER and FER curves of integrated turbo coded systems with code rate 1/2. Puncturing makes the code less powerful than the original rate 1/3 code. Therefore the BER and FER curves shift to the right into higher E_b/N_0 regions. Similar to the curves in Fig. 6.2 and Fig. 6.3, the system that estimates symbol timing with perfect frame synchronization has BER and FER performance close to the performance of systems

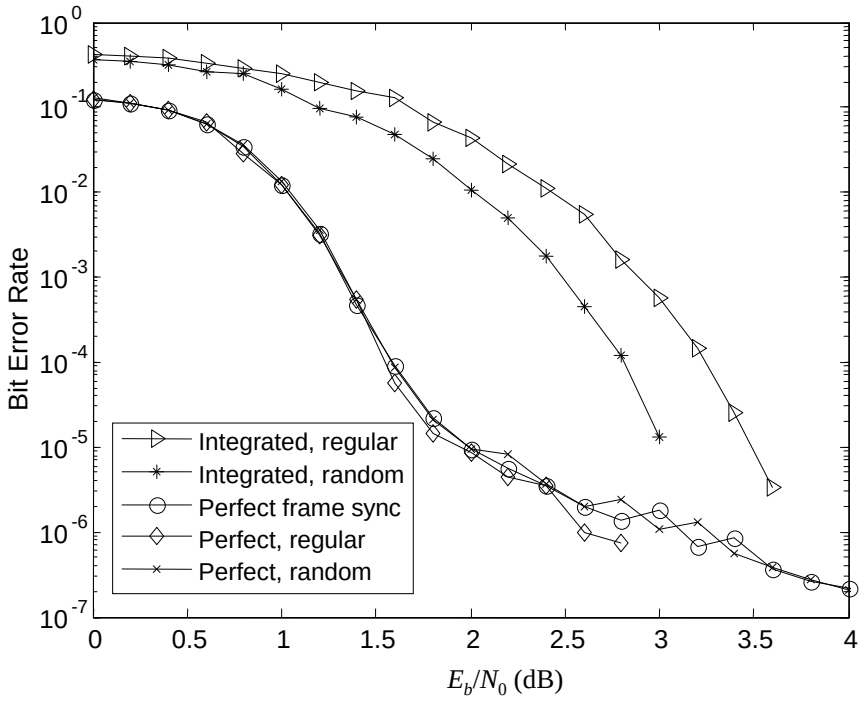


Figure 6.4: Bit error rate of turbo codes with code rate 1/2, interleaver size = 1024. Random interleavers are used. A regular or random scrambler is used. BPSK modulation with raised-cosine roll-off pulse shaping is used with $\alpha = 0.5$.

with perfect symbol timing and frame synchronization.

Puncturing not only increases the code rate, but also increases the density of the parity-check matrix of codes. As stated in Chapter 5, the proposed MAP frame synchronizer suffers from the increased density of the parity-check matrix. It is shown that the gap between the BER curves of perfect synchronization and estimated synchronization is wider than 1 dB in Fig. 6.4 at the BER of 10^{-4}. The random scrambler improves the performance of the system with a regular scrambler by about 0.5 dB. In Fig. 6.5, the FER curve of the integrated systems with a random scrambler is less than 1 dB away from the curve of perfect synchronization. It converges to the curve of perfect synchronization when $E_b/N_0 > 3$ dB.

It is found that frame synchronization is more critical to the BER performance of the integrated systems than symbol timing synchronization. This happens because the decoder

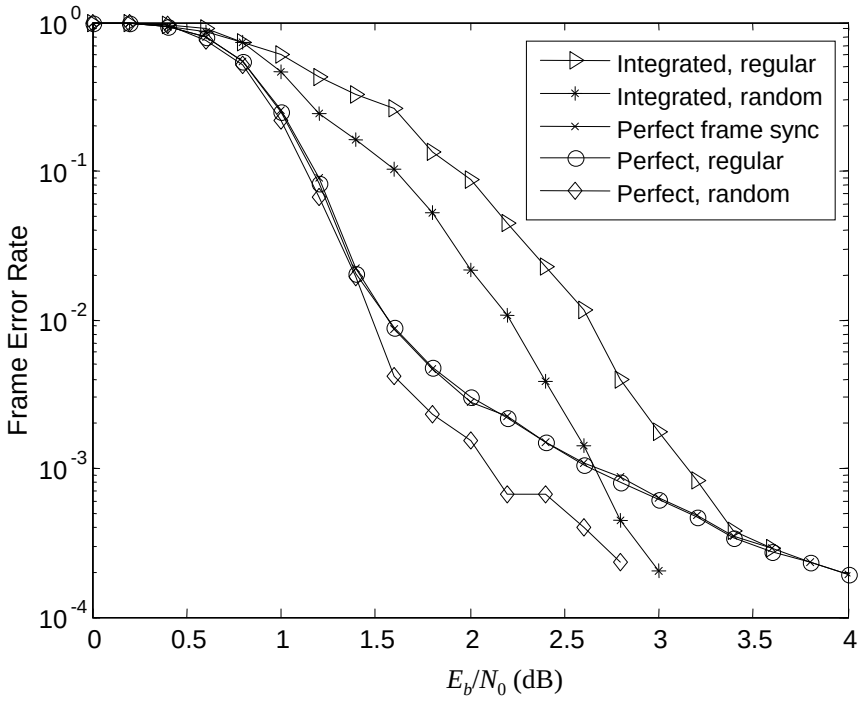


Figure 6.5: Frame error rate of turbo codes with code rate 1/2, interleaver size = 1024. Random interleavers are used. A regular or random scrambler is used. BPSK modulation with raised-cosine roll-off pulse shaping is used with $\alpha = 0.5$.

tries to decode the received codeword even if the frame synchronizer fails to give a reliable decision. Therefore if a codeword is misaligned, then random errors are present in the entire codeword. This problem may be solved by apply automatic request for retransmission (ARQ).

6.4 Future work

6.4.1 Insertion/deletion channel

In a practical communication system, the sampler may skip a symbol or sample one symbol more than required times due to unexpected timing jitter. This causes insertion or deletion of symbols. Research in this dissertation considers the situation that each symbol in a

codeword is received in ideal order, where no insertion or deletion happens. It is of interest to test techniques proposed in this dissertation and/or create new techniques to cope with insertion and deletion channels.

6.4.2 Cycle slips

Chapter 4 shows that random time walk may cause great performance degradation. The degradation is caused by cycle slips. The symbol timing synchronizer must be designed to track cycle slips and mitigate their effect. A code structure capable of dealing with cycle slips may be required to tackle this problem.

Appendix A

List of Symbols

Symbol	Definition
α	Roll-off factor of raised-cosine roll-off function
β	Effective SNR
δ	Gate width of the early late gates
ϵ	Estimation timing error
γ_1	Coefficient of linear time walk
γ_2	Coefficient of random time jitter
γ_3	Residue random time jitter
$\mathbf{\Gamma}$	Log-likelihood ratio of data bits
$\mathbf{\Gamma}^2$	Log-likelihood ratio of data bits from the lower decoder
Γ_i	Log-likelihood ratio of ith bit
κ	Constant related to row weight
$\nu(\mu)$	Parity-check metrics
ω	Feedback scale factor
ρ	Correlation coefficient of online statistics
ρ'	Correlation coefficient of colored noise variances

ψ	Time delay modification produced by M & M TED
σ^2	Variance of additive noise
σ_t^2	Variance of random jitter
$\sigma_{\hat{\tau}}^2$	Variance of estimation errors
τ	Time delay
$\hat{\tau}$	Estimated time delay
τ_0	Initial time delay
τ_1	Linear time walk
τ_2	Accumulated random time jitter
τ_s	Symbol time delay
$a(t)$	Channel gain in the continuous domain
$\mathbf{c}$	Binary code word
$\hat{\mathbf{c}}$	Hard decision on received codeword
C	Channel capacity
$\mathcal{C}$	The set of all valid codewords in GF(2)
$\tilde{\mathcal{C}}$	The modulated version of $\mathcal{C}$
$C(\beta)$	Slope of effective SNR
$C_{DA}(\beta)$	Slope of decision-aided estimation results of effective SNR
c_j	The jth variable node in a Tanner's graph
$\mathbf{c}_i$	Neighborhood of the i-th check node
d_k	Transmitted antipodal data symbols
d_k^i	Soft decision of k-th symbol in the i-th decoding iteration
d_{min}	Minimum distance
$\mathbf{e}$	Error vector
$e(\tau, \beta)$	Error function

e_k^i	Error signal produced by M & M TED
$\mathbf{E}$	Expected value
E_s	Energy per symbol
E_b	Energy per bit
f_i	The ith check node in a Tanner's graph
$g(t)$	Pulse shape function
$\mathbf{G}$	Generating matrix
$\mathbf{G}(D)$	Generating polynomial of a convolutional code
$g_T()$	Transmission pulse shape function
$g_R(t)$	Receive pulse shape function
$G_R(f)$	Fourier transform of $g_R(t)$
$\mathbf{H}$	Parity-check matrix
$\mathbf{H}(D)$	Parity-check polynomial of a convolutional code
$\mathcal{H}_0$	Null hypothesis of frame synchronization
$\mathcal{H}_1$	Alternative hypothesis of frame synchronization
$\mathbf{I}$	Identical matrix
K	Interleaver size of turbo codes
k_c	Constraint length
l	Normalized observation window size
L	Codeword length
$\mathbf{L}_e^i$	Input extrinsic information
$\mathbf{L}_e^o$	Output extrinsic information
L_B	Decision feedback observation window size
L_F	Feedforward observation window size
L_k^i	Log-likelihood ratio of the k-th symbol in the i-th decoding iteration

$L(\mu)$	(Log-)likelihood function for frame synchronization
$M(\tau)$	Mean square error caused by time delay
m_c	Mean of Q_i when there are an even number of 1's in $\mathbf{c}_i$
m_k	Autocorrelation of random data
M_w	Length of the sync word
N	The number of samples per symbol
$\mathcal{N}(m, \sigma^2)$	Normal distribution with mean m and variance σ^2
N_0	One-sided power spectrum density of additive noise
n_k	Data word length
n_m	Number of parity check bits
n_n	Code word length
p_j	Probability of $c_j = 1$
q_i	Probability of receiving an even number of 1's in $\mathbf{c}_i$
Q_i	Logarithm likelihood of receiving an even number of 1's in $\mathbf{c}_i$
r	Code rate
R	Data rate
$r(t)$	Matched filter output
$r[n]$	Sampled matched filter output
$R_w(\nu)$	Auto-correlation of colored additive noise
s_n	Online statistics of the nth sample sequence
s_k^i	Interpolation result of the k-th symbol in the i-th decoding iteration
T	Symbol duration
$\mathbf{v}$	Parity-check bits
$\mathbf{V}$	Variance
$\mathbf{w}$	Noise vector

$w(t)$	Additive noise
W_c	Column weight
w_n	Weight of the nth sample
W_r	Row weight
W_s	The sync word
$w_M(t)$	Matched filter output additive noise
$\mathbf{x}$	Binary data word vector
$\hat{\mathbf{x}}$	Hard decision on $\mathbf{x}$
X	The Fourier transform result of $x[n]$
$x[n]$	Square of samples $r[n]$
$x(t)$	The transmitted signal
x_i	Binary data bit
$\hat{x}_k$	Hard decision for x_i
$y(t)$	Received signal
$\mathbf{z^1}$	Parity bits generated by the upper RSC encoder in a turbo encoder
$\mathbf{z^2}$	Parity bits generated by the lower RSC encoder in a turbo encoder

Appendix B

List of Abbreviations

Abbreviation	Definition
AGC	Automatic gain control
APP	a posteriori probability
appro.	Approximately
ARQ	Automatic request for retransmission
AWGN	Additive white Gaussian noise
BCH	Bose & Chaudhuri & Hochquenghem
BCJR	Bahl & Cocke & Jelinek & Raviv
BER	Bit-error rate
BPSK	Binary phase-shift keying
CSI	Channel state information
DF	Decision-feedback
DSP	Digital signal processing
FEC	Forward error correction
FER	Frame error rate
FPGA	Field programable gate array

i.i.d.	independent, identically distributed
ISI	Inter-symbol interference
ITR	Interpolation timing recovery
LDPC	Low-density parity-check (codes)
LLR	log-likelihood ratio
LMSE	Least mean square error
M & M	Mueller and Müller's
MLSE	Maximum likelihood sequence estimation
MSE	Mean square error
MAP	Maximum a posteriori probability
ML	Maximum likelihood
NDA	Non-data aided
NSC	Non-systematic convolutional (codes)
NTSC	National Television Systems Committee
OFDM	Orthogonal frequency division multiplexing
PAL	Phase Alternating Line
PCCC	Parallel concatenated convolutional codes
P/S	Parallel-to-serial convertor
RC	Raised-cosine
RDL	Random data limit
RF	Radio frequency
RSC	Recursive systematic convolutional (codes)
SISO	Soft-input soft-output
SNR	Signal-to-Noise ratio
STC	Space-time codes

TDMA	Time division multiple access
TED	Time error detector
VA	Viterbi algorithm
VLSI	Very large scale integrate (circuits)

Appendix C

BCJR decoding algorithm

The coding structure of a convolutional code can be expressed by a trellis. Considering a binary RSC code, which has a memory size of m, there are 2^m states. In the kth section of the trellis, each state u has 2 paths coming in for $x_k = 0$ and $x_k = 1$ as shown in Fig. C.1(a). Also each state has 2 paths going out for input $x_{k+1} = 0$ and $x_{k+1} = 1$ as in Fig. C.1(b).

(a) Forward transition in a trellis.

(b) Backward transition in a trellis.

Figure C.1: State transition and output values.

The definitions of the variables in Fig. C.1 are

- $S_b^i(u)$ — The state that goes to u when the input $x_k = i$;
- $V_b^i(u)$ — The output associated with the transition $S_b^i(u) \rightarrow u$ when $x_k = i$;

- $S_f^i(u)$ — The state that comes from u when the input $x_{k+1} = i$;

- $V_f^i(u)$ — The output associated with the transition $u \to S_f^i(u)$ when $x_{k+1} = i$.

Also the following definitions are needed

- Y_k — The kth received symbol in a codeword;

- Y_i^j — The symbols in the received codeword from position i to j. Specifically, Y_1^K is the whole codeword;

- S_k — The state in the kth section of the trellis.

The *a posteriori* probability (APP) of a data bit $x_k = i$ on condition of Y_1^N is

$$\Pr\left(x_k = i \left| Y_1^N \right.\right) = \sum_{u=0}^{2^m-1} \lambda_k^i(u) \tag{C.1}$$

where

$$\lambda_k^i(u) = \Pr\left(x_k = i, S_k = u \left| Y_1^K \right.\right) \tag{C.2}$$

$\lambda_k^i(u)$ can be decomposed into two probabilities

$$\lambda_k^i(u) = \alpha_k^i(u)\,\beta_k^i(u) \tag{C.3}$$

where

$$\begin{aligned} \alpha_k^i(u) &= \Pr\left(x_k = i, S_k = u, Y_1^k\right) \\ \beta_k^i(u) &= \Pr\left(Y_{k+1}^K \left| x_k = i, S_k = u \right.\right) \end{aligned} \tag{C.4}$$

$\alpha_k^i(u)$ is obtained recursively by

$$\alpha_k^i(u) = \exp\left(\frac{\hat{x}_k i + \hat{z}_k V_f^i(u)}{\sigma^2/2}\right) \sum_{j=0}^{1} \alpha_{k-1}^i\left(S_b^j(u)\right) \tag{C.5}$$

where the received codeword is divided into symbols $\hat{x}_k$ associated with data bits x_k, and $\hat{z}_k$ associated with parity bits z_k. $\beta_k^i(u)$ can also be found recursively as

$$\beta_k^i(u) = \sum_{j=0}^{1} \beta_{k+1}^j\left(S_f^i(u)\right) \exp\left(\frac{\hat{x}_{k+1} j + \hat{z}_{k+1} V_f^j\left(S_f^i(u)\right)}{\sigma^2/2}\right) \tag{C.6}$$

The decoding algorithm is summarized in below

1. Initialize for $i = 0, 1$

   $\alpha_0^i(S_b^i(0)) = 1$ and $\alpha_0^i(S_b^i(u)) = 0$ for $u \neq 0$;

   $\beta_K^i(S_b^i(0)) = 1$ and $\beta_K^i(S_b^i(u)) = 0$ for $u \neq 0$.

2. After the whole codeword is received, for all states u, $i = 0, 1$ and $k = 1, \ldots, K$, compute $\alpha_k^i(u)$ and $\beta_k^i(u)$ using (C.5) and (C.6) respectively.

3. Compute Λ_k with

$$\Lambda_k = \log\left(\frac{\sum_{u=0}^{2^m-1} \alpha_k^1(u)\,\beta_k^1(u)}{\sum_{u=0}^{2^m-1} \alpha_k^0(u)\,\beta_k^0(u)}\right) \tag{C.7}$$

Let $L_c = 2/\sigma^2$, which is called the channel reliability. If extrinsic information $L_e^i(k)$ about x_k is available, then (C.5) and (C.6) become

$$\alpha_k^i(u) = \exp\left[\left(L_c \hat{x}_k + L_e^i(k)\right) i + L_c \hat{z}_k V_f^i(u)\right] \sum_{j=0}^{1} \alpha_{k-1}^i\left(S_b^j(u)\right) \tag{C.8}$$

and

$$\beta_k^i(u) = \sum_{j=0}^{1} \beta_{k+1}^j \left(S_f^i(u)\right) \exp\left[\left(L_c \hat{x}_{k+1} + L_e^i(k+1)\right) j + L_c \hat{z}_{k+1} V_f^j \left(S_f^i(u)\right) \sigma^2/2\right]. \quad \text{(C.9)}$$

Appendix D

Message-passing decoding in probability domain

The following denotations are needed

- q_{ij} — messages to be passed from bit node c_i to check nodes f_j.
- r_{ji} — messages to be passed from check node f_j to bit node c_i.
- $R_j = \{i : h_{ji} = 1\}$ — the set of column locations of the 1's in the jth row
- $R_{j\backslash i} = \{i' : h_{ji'} = 1\} \setminus \{i\}$ — the set of column locations of the 1's in the jth row, excluding location i.
- $C_j = \{i : h_{ji} = 1\}$ — the set of row locations of the 1's in the ith column
- $C_{i\backslash j} = \{i' : h_{j'i} = 1\} \setminus \{j\}$ — the set of row locations of the 1's in the ith column, excluding location j.
- $p_i = \Pr(c_i = 1|y_i)$

The decoding algorithm is summarized below.

Compute for $\forall i, j$ that satisfies $h_{ij} = 1$.

1. Initialize

$$\begin{aligned} q_{ij}(0) &= 1 - p_i = \Pr(c_i = 0 \,|y_i) = \frac{1}{1 + e^{-2y_i/\sigma^2}} \\ q_{ij}(1) &= p_i = \Pr(c_i = 1 \,|y_i) = \frac{1}{1 + e^{2y_i/\sigma^2}} \end{aligned} \tag{D.1}$$

2. First half round iteration

$$\begin{aligned} r_{ji}(0) &= \frac{1}{2} + \frac{1}{2} \prod_{i' \in R_{j \setminus i}} (1 - 2q_{i'j}(1)) \\ r_{ji}(1) &= 1 - r_{ji}(0) \end{aligned} \tag{D.2}$$

3. Second half round iteration

$$\begin{aligned} q_{ij}(0) &= K_{ij}(1 - p_i) \prod_{j' \in C_{i \setminus j}} r_{j'i}(0) \\ q_{ij}(1) &= K_{ij} p_i \prod_{j' \in C_{i \setminus j}} r_{j'i}(1) \end{aligned} \tag{D.3}$$

where constants k_{ij} are selected to ensure

$$q_{ij}(0) + q_{ij}(1) = 1 \tag{D.4}$$

4. Soft decision

$$\begin{aligned} Q_i(0) &= K_i(1 - p_i) \prod_{j \in C_i} r_{ij}(0) \\ Q_i(1) &= K_i p_i \prod_{j \in C_i} r_{ij}(1) \end{aligned} \tag{D.5}$$

where constants k_i are selected to ensure

$$Q_i(0) + Q_i(1) = 1 \tag{D.6}$$

5. Hard decision

$$\hat{c}_i = \begin{cases} 1 & \text{if } Q_i(1) > 0 \\ 0 & \text{elsewhere} \end{cases} \tag{D.7}$$

If $\hat{\mathbf{c}}\mathbf{H}^T = \mathbf{0}$ or number of iterations exceeds limitation then stop, else go to Step 2.

Appendix E

Characteristics of online statistics

The online statistics are calculated using the received signal without knowledge of the data sequence.

$$r = \sqrt{\mathcal{E}_s}x + n \tag{E.1}$$

where $x = \{\pm 1\}$, and n is additive white gaussian noise with two-sided power spectral density σ^2. It is desirable to work with only $|r|$ and r^2 since we do not know what x is.

It is found that $R = |r|$ has probability density function (pdf)

$$f_R(r) = \frac{1}{\sqrt{2\pi}\sigma}\left[e^{-\frac{(r-\sqrt{\mathcal{E}_s})^2}{2\sigma^2}} + e^{-\frac{(r+\sqrt{\mathcal{E}_s})^2}{2\sigma^2}}\right] \qquad r \geq 0 \tag{E.2}$$

The distribution of $U = r^2$ has pdf

$$f_U(u) = \frac{1}{2\sqrt{2\pi\sigma^2 u}}\left[e^{-\frac{(\sqrt{u}-\sqrt{\mathcal{E}_s})^2}{2\sigma^2}} + e^{-\frac{(\sqrt{u}+\sqrt{\mathcal{E}_s})^2}{2\sigma^2}}\right] \qquad u \geq 0 \tag{E.3}$$

The online statistics is computed using (3.10). Assume that the numerator and the

denominator in the upper equation are independent, then we have the expected value

$$\mathrm{E}[s] = \mathrm{E}\left[\frac{1}{K}\sum_{i=1}^{K} r_i^2\right] \mathrm{E}\left[\frac{1}{\left[\frac{1}{K}\sum_{i=1}^{K}|r_i|\right]^2}\right] \tag{E.4}$$

The distribution of U in (E.3) is a non-central chi-square distribution. Therefore $\sum_{i=1}^{K} r_i^2$ is a non-central chi-square distribution with K degrees of freedom. The first and second order statistics of $\frac{1}{K}\sum_{i=1}^{K} r_i^2$ are as below.

$$\mathrm{E}\left[\frac{1}{K}\sum_{i=1}^{K} r_i^2\right] = \sigma^2 + \mathcal{E}_s \tag{E.5}$$

$$\mathrm{V}\left[\frac{1}{K}\sum_{i=1}^{K} r_i^2\right] = \frac{2}{K}\sigma^4 + \frac{4}{K}\sigma^2 \tag{E.6}$$

From the central limit theorem, we know that $\frac{1}{K}\sum_{i=1}^{K}|r_i|$ has (approximately) a normal distribution where

$$m_1 = \mathrm{E}\left[\frac{1}{K}\sum|r_i|\right] = \sigma\sqrt{\frac{2}{\pi}}\exp\left(-\frac{1}{2\sigma^2}\right) + \mathrm{erf}\left(\sqrt{\frac{1}{2\sigma^2}}\right) \tag{E.7}$$

$$\sigma_1^2 = \mathrm{V}\left[\frac{1}{K}\sum|r_i|\right] = \frac{\mathcal{E}_s + \sigma^2}{K} - \frac{1}{K}\left[\sigma\sqrt{\frac{2}{\pi}}\exp\left(-\frac{1}{2\sigma^2}\right) + \mathrm{erf}\left(\sqrt{\frac{1}{2\sigma^2}}\right)\right]^2. \tag{E.8}$$

$Y = \left(\frac{1}{K}\sum_{i=1}^{K}|r_i|\right)^2$ has a non-central chi-square distribution. We need to compute the statistics of $\mathrm{E}\left[\frac{1}{Y}\right]$ and $\mathrm{V}\left[\frac{1}{Y}\right]$. It is not possible to give a closed form expression for $\mathrm{E}\left[\frac{1}{Y}\right]$ and $\mathrm{E}\left[\frac{1}{Y^2}\right]$. However, when $\sigma_1^2 << 1$, $\mathrm{E}\left[\frac{1}{Y}\right]$ approaches $(1/m_1^2 + \sigma_1^2)$, and $\mathrm{V}\left[\frac{1}{Y}\right]$ converges to $(2\sigma_1^4 + 4\sigma_1^2)$. If K is sufficiently large, then $\sigma_1^2 << 1$. Hence the expected value and

variance of the online statistic is

$$\begin{aligned} \mathrm{E}[s] &= \left(\mathcal{E}_s + \sigma^2\right)\left(\left[\frac{1}{\sigma\sqrt{\frac{2}{\pi}}\exp\left(-\frac{1}{2\sigma^2}\right) + \operatorname{erf}\left(\sqrt{\frac{1}{2\sigma^2}}\right)}\right]^2\right) \\ &+ \left(\mathcal{E}_s + \sigma^2\right)\left(\frac{\mathcal{E}_s + \sigma^2}{K} - \frac{1}{K}\left[\sigma\sqrt{\frac{2}{\pi}}\exp\left(-\frac{1}{2\sigma^2}\right) + \operatorname{erf}\left(\sqrt{\frac{1}{2\sigma^2}}\right)\right]^2\right) \end{aligned} \quad \text{(E.9)}$$

$$\begin{aligned} \mathrm{V}[s] &= \left[\frac{2\sigma^4}{K} + \frac{4\sigma^2}{K} + \left(\sigma^2 + \mathcal{E}_s\right)^2\right]\left(4\sigma_1^2 + 2\sigma_1^4\right) \\ &+ \left(\frac{2\sigma^4}{K} + \frac{4\sigma^2}{K}\right)\left\{\left[\frac{1}{\sigma\sqrt{\frac{2}{\pi}}\exp\left(-\frac{1}{2\sigma^2}\right) + \operatorname{erf}\left(\sqrt{\frac{1}{2\sigma^2}}\right)}\right]^2 + \sigma_1^2\right\}. \end{aligned} \quad \text{(E.10)}$$

Bibliography

[1] J. G. Proakis, *Digital communications*. McGraw-Hill, Inc., 4th ed., 2000.

[2] T. S. Rappaport, *Wireless Communications: Principles and Practice*. Upper Saddle River, NJ: Prentice Hall PTR, 1996.

[3] J. K. Cavers, "An analysis of pilot symbol assisted modulation for Rayleigh fading channels," *IEEE Trans. Veh. Tech.*, vol. 40, pp. 686–693, Nov. 1991.

[4] J. B. Thomas, *An Introduction to Statistical Communication Theory*. Wiley, New York, 1969.

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

[6] F. M. Gardner, "A BPSK/QPSK timing-error detector for sampled receivers," *IEEE Trans. Comm.*, vol. COM-34, pp. 423–429, May 1986.

[7] H. Meyr, M. Moeneclaey, and S. Fechtel, *Digital communication receivers: synchronization, channel estimation and signal processing*. John Wiley and Sons, Inc., 1998.

[8] M. Oerder and H. Meyr, "Digital filter and square timing recovery," *IEEE Trans. Comm.*, vol. 36, pp. 605–612, May 1988.

[9] M. Morelli, A. N. D'Andrea, and U. Mengali, "Feedforward ML-based timing estimation with PSK signals," *IEEE Communications Letters*, vol. 1, pp. 80–82, May 1997.

[10] S. J. Lee, "A new non-data-aided feedforward symbol timing estimator using two samples per symbol," *IEEE Communications Letters*, vol. 6, pp. 205–207, May 2002.

[11] U. Mengali and A. N. D'Andrea, *Synchronization Techniques for Digital Receivers*. Plenum press, 1997.

[12] Y. Wang, E. Serpedin, and P. Ciblat, "An alternative blind feedforward symbol timing estimator using two samples per symbol," *IEEE Trans. Comm.*, vol. 51, pp. 1451–1455, Sep. 2003.

[13] B. Mielczarek and A. Svensson, "Timing error recovery in turbo coded systems on AWGN channels," *IEEE Trans. Comm.*, vol. 50, pp. 1584–1592, October 2002.

[14] J. Sun and M. C. Valenti, "Synchronization of turbo codes based on online statistics," in *Proc. IEEE Inter. Conf. Comm.*, vol. 4, (Anchorage AK), pp. 2733–2737, May 2003.

[15] J. G. Proakis and D. G. Manolakis, *Digital Signal Processing: Principles, Algorithms, and Applications*. New York: Maxwell Macmillan International,, 2 ed., 1992.

[16] R. A. Scholtz, "Frame synchronization techniques," *IEEE Trans. Comm.*, vol. 28, pp. 1204–1213, Aug. 1980.

[17] J. L. Massey, "Optimum frame synchronization," *IEEE Trans. Comm.*, vol. 20, pp. 115–119, Apr. 1972.

[18] G. L. Lui and H. H. Tan, "Frame synchronization for Gaussian channels," *IEEE Trans. Comm.*, vol. COM-35, pp. 818–829, August 1987.

[19] P. Robertson, "A generalized frame synchronizer," in *Proc. Global Telecommunications Conf.*, vol. 1, pp. 365–369, Dec. 1992.

[20] P. Robertson, "Optimum frame synchronization of packets surrounded by noise with coherent and differentially coherent demodulation," in *Proc. International Conf. Comm.*, pp. 874–879, May 1994.

[21] M. M. K. Howlader and B. D. Woerner, "Frame synchronization of convolutionally encoded sequences for packet transmission," in *Proc. IEEE ICC*, (New Orleans), pp. 317–321, June 2000.

[22] M. M. K. Howlader and B. D. Woerner, "Decoder-assisted frame synchronization for packet transmission," *IEEE Journal on Selected Areas in Communications*, vol. 19, pp. 2331–2345, Dec. 2001.

[23] T. M. Cassaro and C. N. Georghiades, "Frame synchronization for coded systems over AWGN channels," *IEEE Trans. Comm.*, vol. 32, pp. 484–489, Mar. 2004.

[24] W. Matsumoto and H. Imai, "Blind synchronization with enhanced sum-product algorithm for low density parity-check codes," in *International Symposium on Wireless Personal Multimedia Communications*, vol. 3, pp. 966–970, Oct. 2002.

[25] S.-Y. Chung, T. J. Richardson, and R. L. Urbanke, "Analysis of sum-product decoding of low-density parity-check codes using a Gaussian approximation," *IEEE Trans. Info. Theo.*, vol. 47, pp. 657–670, Feb. 2001.

[26] J. Sun and M. C. Valenti, "Joint synchronization and SNR estimation for turbo codes in AWGN channels." Accepted, IEEE Trans. Comm.

[27] J. Sun and M. C. Valenti, "Mitigation of random time walk in turbo coded systems." Submitted, IEEE VTC Spring, 2005.

[28] J. Sun and M. C. Valenti, "Optimum frame synchronization for preamble-less packet transmission of turbo codes," Nov. 2004. Accepted, Asilomar Conference on Signals, Systems, and Computers.

[29] C. E. Shannon, "A mathematical theory of communication," *Bell Sys. Tech. J.*, vol. 27, pp. 379–423 and 623–656, 1948.

[30] C. Berrou, A. Glavieux, and P. Thitimasjshima, "Near Shannon limit error-correcting coding and decoding: Turbo-codes(1)," in *Proc., IEEE Int. Conf. on Commun.*, (Geneva, Switzerland), pp. 1064–1070, May 1993.

[31] D. MacKay, "Good error correcting codes based on very sparse matrices," *IEEE Trans. Information Theory*, pp. 399–431, March 1999.

[32] S. Lin and J. Daniel J. Costello, *Error Control Coding : Fundamentals and Applications*. Englewood Cliffs, NJ: Prentice-Hall, 1983.

[33] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Info. Theory*, vol. 13, pp. 260–269, Apr. 1967.

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

[35] G. D. Forney, Jr., *Concatenated codes*. MIT Press, 1966.

[36] S. A. Barbulescu, *Iterative Decoding of Turbo Codes and Other Concatenated Codes*. PhD thesis, University of South Australia, Feb. 1996.

[37] R. Gallager, "Low-density parity-check codes," *IRE Trans. Information Theory*, pp. 21–28, January 1962.

[38] T. J. Richardson and R. L. Urbanke, "The capacity of low-density parity-check codes under message-passing decoding," *IEEE Trans. Info. Theo.*, vol. 47, pp. 599–618, Feb. 2001.

[39] T. J. Richardson and R. L. Urbanke, "Efficient encoding of low-density parity-check codes," *IEEE Trans. Info. Theo.*, vol. 47, pp. 638–656, Feb. 2001.

[40] R. M. Tanner, "A recursive approach to low complexity codes," *IEEE Trans. Info. Theory*, vol. IT-27, pp. 533–547, Sep. 1981.

[41] J. Liu, H. Song, and B. V. K. V. Kumar, "Symbol timing recovery for low-SNR partial response recording channels," in *Proc., IEEE GLOBECOM*, pp. 1129–1136, 2002.

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

[43] A. R. Nayak, J. R. Barry, and S. W. McLaughlin, "Lower bounds for the performance of iterative timing recovery at low SNR," in *15th International Symposium on the Mathematical Theory of Networks and Systems (MTNS)*, (University of Notre Dame, South Bend, Indiana), pp. WM2–5, Aug. 2002.

[44] Z.-N. Wu, J. M. Cioffi, and K. D. Fisher, "A MMSE interpolated timing recovery scheme for the magnetic recording channel," in *IEEE Int. Conf. on Commun.*, vol. 3, (Montreal), pp. 1625–1629, June 8 - 12 1997.

[45] L. Lu and S. G. Wilson, "Synchronization of turbo coded modulation systems at low SNR," in *Proc., IEEE Int. Conf. on Commun.*, pp. 428–432, 1998.

[46] R. V. Hogg and A. T. Craig, *Introduction to mathematical statistics.* Prentice Hall, Inc., 5th ed., 1995.

[47] C. N. Georghiades and M. Moeneclaey, "Sequence estimation and synchronization from nonsynchronized samples," *IEEE Trans. Info. Thoery*, vol. 37, pp. 1649–1657, Nov. 1991.

[48] L. E. Franks, "Further results on Nyquist's problem in pulse transmission," *IEEE Trans. Comm. Tech.*, vol. 16, pp. 337–340, April 1968.

[49] Third Generation Partnetship Project 2 (3GPP2), "Physical layer standard for cdma2000 spread spectrum systems, release c," in *3GPP2 C.S0002-C*, pp. 115–122, May 2002. available online at http://www.3gpp2.org.

[50] T. A. Summers and S. G. Wilson, "SNR mismatch and online estimation in turbo decoding," *IEEE Trans. Comm.*, vol. 46, pp. 421–423, April 1998.

[51] L. Erup, F. M. Gardner, and R. A. Harris, "Interpolation in digital modems – part II: implementation and performance," *IEEE Trans. Comm.*, vol. 41, pp. 998–1008, June 1993.

[52] M. Reed and J. Asenstorfer, "A novel variance estimator for turbo-code decoding," in *Proc., Int. Conf. Telecom.*, (Melbourne, Australia), pp. 173–178, April 1997.

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

[54] I. V. Kozintsev, "http://www.kozintsev.net/soft.html." Personal Web Page, May 2004.

[55] K. Engdahl and K. S. Zigangirov, "On the theory of low-density convolutional codes I," in *Probl. Peredach. Inform.*, vol. 35, pp. 12–28, Oct.-Nov.-Dec. 1999.

[56] G. D. Forney, Jr., "Structural analysis of convolutional codes via dual codes," *IEEE Trans. Info. Theory*, vol. IT-19, pp. 512–518, July 1973.

[57] R. Johnanesson and K. S. Zigangirov, *Fundamentals of convolutional coding*. IEEE Press series on digital and mobile communication, IEEE Press, 1998.

Contributions

1. M. C. Valenti and J. Sun, "The UMTS turbo code and an efficient decoder implementation suitable for software-defined radios," *International Journal of Wireless Information Networks*, vol. 8, pp. 203–215, Oct. 2001.

2. J. Sun and M. C. Valenti, "Synchronization of turbo codes based on online statistics," in *Proc. IEEE Inter. Conf. Comm.*, vol. 4, (Anchorage AK), pp. 2733–2737, May 2003.

3. J. Sun and M. C. Valenti, "Turbo codes.", Chap. 12 in *Handbook of RF and wireless technologies*, Editor: F. Dowla, Newnes, 2004.

4. J. Sun and M. C. Valenti, "Joint synchronization and SNR estimation for turbo codes in AWGN channels." submitted to IEEE Trans. Comm.

5. J. Sun and M. C. Valenti, "Optimum frame synchronization for preamble-less packet transmission of turbo codes," Nov. 2004. Accepted, Asilomar Conference on Signals, Systems, and Computers.

6. J. Sun and M. C. Valenti, "Mitigation of random time walk in turbo coded systems." Submitted, IEEE VTC Spring, 2005.