Iterative Decoding and Channel Estimation over Hidden Markov Fading Channels

Anwer A. Khan

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

William J. Ebel, Chair
William H. Tranter
Charles W. Bostian
F. Gail Gray

May 3, 2000
Blacksburg, Virginia

Keywords: Iterative Decoding, Channel Estimation, Hidden Markov Models, Fading Channels
Copyright 2000, Anwer A. Khan

Iterative Decoding and Channel Estimation over Hidden Markov Fading Channels

Anwer A. Khan

(ABSTRACT)

Since the 1950s, hidden Markov models (HMMS) have seen widespread use in electrical engineering. Foremost has been their use in speech processing, pattern recognition, artificial intelligence, queuing theory, and communications theory. However, recent years have witnessed a renaissance in the application of HMMs to the analysis and simulation of digital communication systems. Typical applications have included signal estimation, frequency tracking, equalization, burst error characterization, and transmit power control. Of special significance to this thesis, however, has been the use of HMMs to model fading channels typical of wireless communications. This variegated use of HMMs is fueled by their ability to model time-varying systems with memory, their ability to yield closed form solutions to otherwise intractable analytic problems, and their ability to help facilitate simple hardware and/or software based implementations of simulation test-beds.

The aim of this thesis is to employ and exploit hidden Markov fading models within an iterative (turbo) decoding framework. Of particular importance is the problem of channel estimation, which is vital for realizing the large coding gains inherent in turbo coded schemes. This thesis shows that a Markov fading channel (MFC) can be conceptualized as a trellis, and that the transmission of a sequence over a MFC can be viewed as a trellis encoding process much like convolutional encoding. The thesis demonstrates that either maximum likelihood sequence estimation (MLSE) algorithms or maximum *a posteriori* (MAP) algorithms operating over the trellis defined by the MFC can be used for channel estimation. Furthermore, the thesis illustrates sequential and decision-directed techniques for using the aforementioned trellis based channel estimators *en masse* with an iterative decoder.

Acknowledgments

I would like to express my gratitude to my advisor, Dr. William J. Ebel, for his advise, support, and encouragement. I wish to extend special thanks to Dr. William H. Tranter and to Dr. Charles W. Bostian for not only serving on my advising committee, but also for being great mentors both in and out of the classroom. I would also like to thank Dr. F. Gail Gray for serving on my advising committee, and to Dr. Jeff H. Reed for his constant encouragement.

I also wish to express my appreciation to my friends at the Center for Wireless Telecommunications (CWT) and at the Mobile and Portable Radio Research Group (MPRG). Special thanks to Yufei Wu for help on turbo codes, James Hicks for help on deciphering the cryptic works of Dr. William Turin, and Roberto Conte and Carl Fossa for their friendship and assistance.

I would also like to thank the staff at CWT and MPRG for their help and assistance. I am also grateful for the generous financial support that I received from the Aerospace Corporation and from Texas Instruments.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The discovery of turbo codes in 1993 was perhaps the most significant breakthrough in the field of channel coding since the introduction of trellis codes in 1982. While trellis codes offer near capacity performance in band-limited channels, turbo codes offer near capacity performance in power-limited channels such as deep space, satellite, and terrestrial wireless channels. Turbo codes have, therefore, attracted a lot of attention, and have been a topic of active research over the last few years. Not only has this research led to an improvement in the design and performance of turbo codes *per se*, it has, more importantly, led to a plethora of baseband applications and advance receiver designs that make use of the iterative decoding or the turbo processing principle. However, the near optimum performance of turbo codes and/or the performance gains realized via iterative decoding is conditioned upon reliable knowledge of the channel side information[1]. Lack of this knowledge can severely degrade the performance of turbo codes and/or offset any gains that may otherwise have been realized via turbo processing. Therefore, the need for reliable channel knowledge necessitates some form of channel estimation at the receiver.

The aim of this thesis is to investigate novel channel estimation and iterative decoding schemes by using a hidden Markov model of the channel. This chapter presents the motivation behind this thesis. Sections (1.1) and (1.2) present a brief description of turbo codes and highlight the significance of the iterative decoding principle in modern receiver designs. Section (1.3) explains the importance of channel estimation to iterative decoding. Section (1.4) describes in detail the purpose and the outline of this thesis.

[1]The channel side information includes parameters such as the channel gain or the fading amplitude, the carrier phase, and the noise variance.

1.1 Turbo Codes

Turbo codes were first brought to light by the French team of C. Berrou, A. Glavieux, and P. Thitimajshima at the International Conference on Communications in June 1993 [3]. In its original form[2], turbo codes consist of a parallel concatenation of two rate 1/2 recursive systematic convolutional (RSC) encoders. The two RSC encoders are separated by an interleaver, such that the first encoder receives the original copy of the input sequence, while the second encoder receives an interleaved or permuted version of the input sequence. The interleaver is, therefore, responsible for the random-like nature of turbo codes. It is this inherent randomness in turbo codes that enables them to perform extremely close to the Shannon limit [1]. For instance, it was shown by Berrou et al in [3] that turbo codes could achieve a bit error rate of 10^{-5} at an E_b/N_o of 0.7 dB. The presence of an interleaver, however, complicates the optimal maximum likelihood decoding of turbo codes. Turbo decoding, therefore, proceeds by breaking the overall decoding problem into two less complicated decoding operations—each RSC code is decoded independently—with locally optimum solutions, and by sharing information in an iterative fashion between the two decoding operations. Each decoder is implemented as a soft-in soft-out (SISO) module, and generates the *a posteriori* probabilities (APPs) of the data bits. Moreover, the two decoders operate in a symbiotic (feedback) manner, where the output of one feeds the input of the other, and *vice versa*. It is this decoding strategy that gives turbo codes their name, since the feedback action of the decoder is reminiscent of a turbo-charged engine.

1.2 Turbo Processing

Modern digital receivers consists of a cascade of signal processing subsystems, such as adaptive antennas, equalizers, multiuser detectors, channel decoders, and source decoders. Each subsystem is usually optimized for a specific task, and is typically incognizant and independent of the other subsystems in the processing chain. These subsystems interface with each other by exchanging bits or hard-decisions between themselves. However, whenever a subsystem makes hard-decisions, information is lost, and becomes unavailable to other subsystems in the processing chain. The interface between these subsystems can be greatly improved by borrowing the iterative decoding strategy from turbo codes. With this strategy—known as turbo processing—each subsystem is implemented as a soft-in soft-out (SISO) module. The soft-decisions from a module are passed down the processing chain, where they are refined by subsequent stages. The soft-output of the final stage is fed back to the first stage, and a second iteration of processing is initiated. This process is repeated until the desired performance criterion is satisfied. With this approach, subsystems at the start of the processing chain can take advantage of the information derived by subsystems at the end of the chain. The net effect is a remarkable improvement in performance over the conventional

[2]The original form refers to the scheme proposed by Berrou et al.

scheme mentioned above. Turbo processing has, therefore, found numerous applications, such as joint channel and source decoding [4], [5], symbol detection [6], equalization [7], [8], antenna diversity [9], multiuser detection [10], [11], [40], interference mitigation [12], [13], and turbo-coded modulation [14], [15], [16], [17].

1.3 Iterative Decoding and Channel Estimation

Recall that a turbo decoder consists of two constituent decoders which operate in an iterative and symbiotic manner. Each constituent decoder is implemented as a soft-in soft-out (SISO) module. A SISO module computes the *a posteriori* probabilities (APPs) of the data bits by accepting as inputs the soft output of the other SISO module, and the channel reliability information. This channel reliability information is derived from the received sequence. For instance, consider a communication system with binary PSK modulation, and a channel model

$$y_i = a_i \mathrm{e}^{j\theta_i}(2x_i - 1) + n_i \tag{1.1}$$

where $x_i \in \{0, 1\}$ is the transmitted symbol, y_i is the received symbol, a_i is the channel gain or the fading amplitude, θ_i is the carrier phase, n_i is a zero-mean Gaussian noise sample with variance $N_o/2E_s$, E_s is the energy per symbol, and $N_o/2$ is the double-sided noise power spectral density. Then, the channel reliability metric for the received symbol, y_i, is given by

$$R_i = 4|a_i \mathrm{e}^{j\theta_i}|\frac{E_s}{N_o}y_i \tag{1.2}$$

Note that the computation of the channel reliability metric necessitates the need for channel estimation or requires the knowledge of the channel side information (SI), viz., the fading amplitude, the carrier phase, and the noise variance. It was shown in [18], [19], and [20] that imperfect channel estimates can considerably alter and degrade the performance of turbo codes. Robust channel estimation is, therefore, paramount in order to realize the large coding gains inherent in turbo coded and/or turbo processed schemes.

1.4 Purpose and Outline of Thesis

Most of the early work on turbo codes assumed perfect knowledge of the channel side information. The observed near optimum performance of turbo codes was, therefore, conditioned upon this perfect knowledge of the channel parameters. However, in practice, these channel parameters need to be estimated at the receiver. Typically, the fading amplitude and the carrier phase are jointly estimated as the complex channel gain[3] using either pilot symbol

[3]If a is the fading amplitude, and θ is the carrier phase, then the complex channel gain, $c = a\mathrm{e}^{\theta}$.

assisted modulation (PSAM) or pilot tone assisted modulation (PTAM) techniques [21], [22]. Likewise, the noise variance is usually estimated by computing the sample variance of the received sequence in a decision-directed mode by using the estimates of the complex channel gain and the output of the turbo decoder [23]. In either case, the ability to reliably estimate the fading amplitude (or the complex channel gain) is of paramount importance.

The ease and reliability with which the fading amplitudes are estimated depends upon the type of model used to represent and characterize the channel. Recent years have seen a growing interest in the use of Markov models to represent communication channels, especially fading channels that are typical of wireless communications. Not only do Markov models enable us to construct discrete fading channel models that can easily be implemented and simulated on computer hardware and/or software, they facilitate (as we shall see) simple trellis based solutions to the otherwise complicated task of estimating the fading amplitudes. For instance, Markov models allow us to compute optimum maximum *a posteriori* estimates of the fading amplitudes using well known techniques such as the Viterbi algorithm and/or the forward-backward algorithm.

The aim of this thesis is to investigate the aforementioned novel approach to channel estimation and iterative decoding by using a hidden Markov model of the fading channel. We begin by taking an in-depth look at turbo codes. Chapter (2) describes in detail the turbo encoding and decoding process. It provides a theoretical analysis of the performance of turbo codes and explains the significance of the constituent elements that make up the turbo encoder and the turbo decoder.

Chapter (3) describes the process of constructing Markov models for fading channels. The chapter provides an introduction to finite state channels, hidden Markov models, and more importantly, to Markov fading channels. It explains the advantages of a discrete Markov fading model *vis-a-vis* the ubiquitous classical (waveform) model. It describes in detail several techniques used for estimating Markov models, and compares the performance of each technique using results from computer simulations.

The advantages of a Markov fading channel are further highlighted in Chapter (4). This chapter develops the maximum *a posteriori* (MAP) criterion for estimating the fading amplitudes. It derives both sequence estimation (Viterbi) and symbol-by-symbol estimation (forward-backward) algorithms for channel estimation. The chapter provides a simple theoretic analysis for evaluating the performance of the MAP estimator, and substantiates the theoretical inferences with results from computer simulations.

Chapter (5) consolidates the work presented in the previous three chapters. It develops several iterative decoding strategies using Markov fading channels and the two aforementioned trellis based channel estimators. The performance of each iterative decoding strategy is benchmarked using computer simulations, compared to one another, and compared against the ideal case of perfect channel cognizance.

Chapter 2

The Turbo Codec

Ever since Shannon's original paper in 1948, coding theorist have attempted to construct structured codes to achieve channel capacity. Structured codes help facilitate practical implementations of encoding and decoding algorithms. However, they perform significantly inferior to the random coding bounds predicted by Shannon. The goal of achieving channel capacity remained unfulfilled until 1993, when a team of French researchers discovered *turbo codes*. Although structured, turbo codes possess random-like properties that enables them to perform extremely close to the Shannon bound. Turbo codes have, therefore, attracted a large group of researchers worldwide, who since 1993 have enhanced its performance, and extended its application to many other areas in communications.

The aim of this chapter is to present a survey of the turbo encoding and decoding scheme. Sections (2.1) and (2.2) present Shannon's channel coding theorem, and an introduction to turbo codes. Sections (2.3) and (2.4) describe the turbo encoding process, and the performance characteristics of turbo codes. Section (2.5) explains the turbo decoding principle, while Section (2.6) describes the different decoding algorithms.

2.1 Noisy Channel Coding Theorem

Consider the communication system shown in Figure (2.1). The information symbol m of dimension k is encoded into an n dimensional codeword X, suitable for transmission over a channel with capacity C[1]. Hence, the rate of the code is $r = k/n$. The received version of the codeword, Y, is passed through a channel decoder, which produces an estimate $\hat{m}$ of the information symbol.

Shannon's *Noisy Channel Coding Theorem* states that it is possible to achieve reliable com-

[1]The channel capacity represents the number of information bits per channel use (or per second) that can be transmitted with an arbitrarily low error rate over a channel.

Figure 2.1: A general communication system.

munications over a channel (with as small an error probability as desired) if the code rate $r < C$ [1], i.e.,

$$\lim_{T\to\infty} \frac{\log N(T,q)}{T} = C \tag{2.1}$$

where $N(T,q)$ is the maximum codebook size of duration T, and q is the error probability such that q does not equal 0 or 1. Shannon intuitively justified the achievability part of the aforementioned statement by introducing the concept of random encoding. The error probability is averaged apropos the codebook choice, and is shown to vanish asymptotically with T if the code rate is less than C. However, the proof is non-constructive, i.e., although the theorem proves the existence of *good* codes, it does not specify which codes can or cannot achieve channel capacity.

Equally important is the converse channel coding theorem which follows directly from Fano's inequality [2], [24]. The converse theorem states that it is impossible to achieve a low error probability, regardless of the type of code used if $r > C$.

Pursuant to these theorems, long random codes with $r < C$ yield optimum performance. However, random codes are not practical to implement, since they lead to extremely complicated (or unrealizable) encoding and decoding algorithms. Therefore, the challenge for coding theorist has always been (and continues to be) to design *random-like* codes which have efficient decoding algorithms.

2.2 Turbo Codes

Coding theorist have traditionally attacked the problem of designing good codes by developing codes with a lot of structure. This approach lends itself to the design of feasible encoders and decoders. However, it results in codes whose performance is significantly inferior to the random coding bounds predicted by Shannon [1]. Turbo codes also contain some structure that facilitates practical implementation, but unlike other coding schemes, they possess random-like properties that allows them to perform extremely close to Shannon's capacity limit. For instance, Berrou[2] et al [25] showed that turbo codes could achieve a bit error rate of 10^{-5} at an E_b/N_o of 0.7 dB. This remarkable energy efficiency has made turbo codes

[2]The French team of C. Berrou, A. Glavieux, and P. Thitimajshima discovered turbo codes in 1993.

extremely attractive for use over power limited and/or interference limited channels. For instance, turbo codes are especially suited for deep space communications [26], [27], [28], satellite communications [29], [30], [31], and terrestrial wireless communications [32], [33], [34], [35], [36].

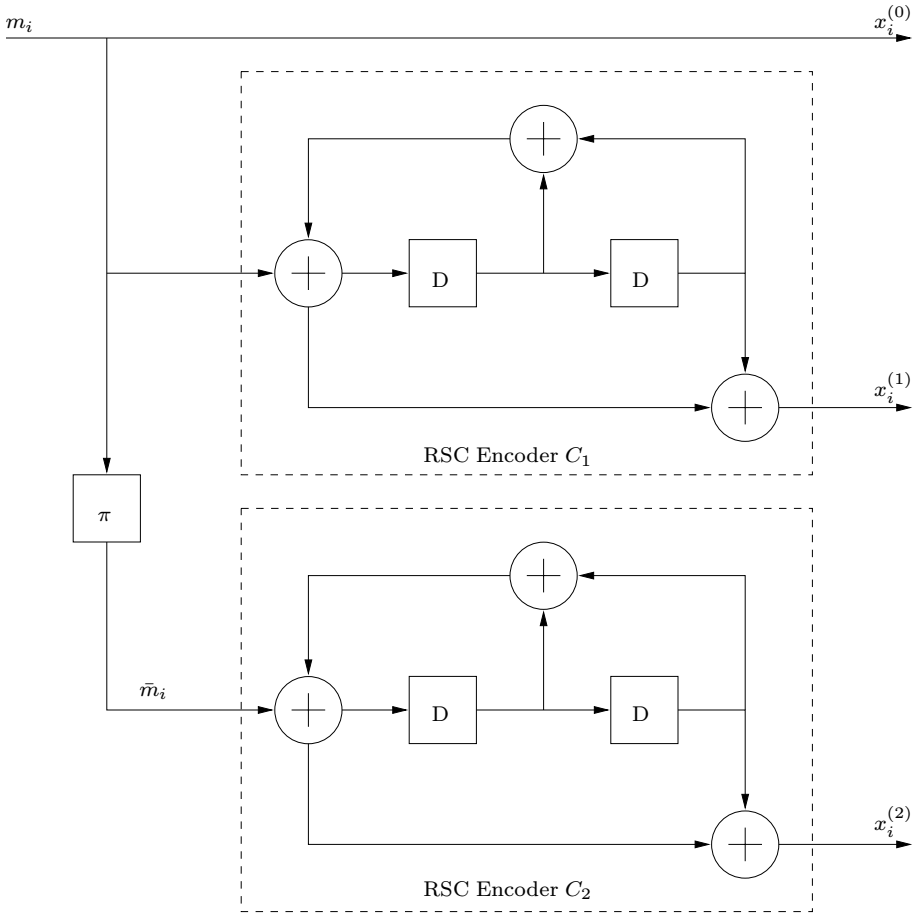

Figure 2.2: A rate 1/3 turbo encoder.

2.3 Turbo Encoding

The *classical* turbo encoding scheme proposed by Berrou et al [25] consists of a parallel concatenation of two recursive systematic convolutional (RSC) codes. Figure (2.2) shows the block diagram of such a turbo encoder. The two rate 1/2 RSC encoders, C_1 and C_2, are separated by an interleaver π. An information symbol m_i is fed directly into C_1. Its interleaved (permuted) version, $\bar{m}_i$, is fed into C_2. The systematic output of the turbo encoder, $x_i^{(0)}$, is taken from C_1, while the two parity outputs, $x_i^{(1)}$ and $x_i^{(2)}$, are taken from

C_1 and C_2. These three outputs are multiplexed to form the codeword $\mathbf{x} = \{x_i^{(0)}, x_i^{(1)}, x_i^{(2)}\}$. Hence, the overall code rate of the turbo encoder is 1/3. However, other code rates can be derived by using an appropriate puncturing[3] scheme. For instance, a rate 1/3 turbo code can be transformed into a rate 1/2 turbo code by using the puncturing matrix

$$P_M = \begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{2.2}$$

where the two parity outputs are alternately punctured. In general, two or more RSC codes can be concatenated in parallel to obtain multiple turbo codes [28], in series to obtain serial turbo codes [37], or in an arbitrary topology composed of both parallel and serial concatenations [38]. Furthermore, the constituent RSC codes can also be different [39]. In this thesis, however, we shall only consider the classical turbo encoding scheme.

2.3.1 Recursive Systematic Convolutional Codes

Consider a rate k/n convolutional code, where k and n are the number of input and output streams of the encoder. The information stream, $\mathbf{m}$, is split into k streams $m^{(p)} = \{m_p, m_{p+k}, m_{p+2k}, \dots\}$, where $p \in \{0, \dots, k-1\}$. Each of these k streams are passed through the convolutional encoder, which produces n output streams, $x^{(q)}$, where $q \in \{0, \dots, n-1\}$. These n output streams are multiplexed to form the codeword $\mathbf{x} = \{x_0^{(0)}, \dots, x_0^{(n-1)}, x_1^{(0)}, \dots, x_1^{(n-1)}, \dots\}$. There are, therefore, k shift registers in the encoder, one for each input stream. Let M_p be the length of the shift register associated with the input p. Then, the total memory of the encoder is given by

$$M_c = \sum_{p=0}^{k-1} M_p \tag{2.3}$$

Furthermore, the constraint length[4] of the encoder is given by

$$K_c = 1 + \max_p \{M_p\} \tag{2.4}$$

The encoding process can be described as the convolution of the input with the coefficients of the generator polynomial (tap weights of the shift register), i.e.,

$$v_i^{(p,q)} = \sum_{j=0}^{M_p} m_{i-j}^{(p)} g_j^{(p,q)} \qquad i \in \{0, \dots, N_b - 1\} \tag{2.5}$$

[3]Puncturing is the process of systematically deleting bits from the output stream of an encoder.

[4]Constraint length is the maximum number of bits in a single output stream that can be affected by any input bit.

where $g_j^{(p,q)}$ is the j-th coefficient of the generator polynomial associated with input p and output q, and N_b is the length of the input sequence. The output stream is formed by

$$\mathbf{x}^{(q)} = \sum_{p=0}^{k-1} \mathbf{v}^{(p,q)} \tag{2.6}$$

A recursive systematic convolutional (RSC) code can be obtained form the aforementioned convolutional (non-recursive, non-systematic) code by using a feedback loop, and by setting one (or more) of the output streams equal to one (or more) input streams. For instance, consider a rate $1/n$ convolutional code with generator matrix (for a rate $1/n$ code, p equals 0, and is discarded from all subsequent notations)

$$g(D) = \left[g^{(0)}(D)\ g^{(1)}(D)\ \cdots\ g^{(n-1)}(D)\right] \tag{2.7}$$

To obtain the equivalent RSC code, we compute the remainder polynomial

$$r(D) = \frac{m(D)}{g^{(0)}(D)} \tag{2.8}$$

where $m(D)$ is the information or message polynomial. Consequently, the parity outputs are given by

$$x^{(q)}(D) = r(D)g^{(q)}(D) \qquad q \in \{1, \ldots, n-1\} \tag{2.9}$$

and the systematic output is simply

$$x^{(0)}(D) = m(D) \tag{2.10}$$

Hence, the generator matrix for the RSC code can be written as

$$g_{RSC}(D) = \left[1\ \frac{g^{(1)}(D))}{g^{(0)}(D)}\ \cdots\ \frac{g^{(n-1)}(D))}{g^{(0)}(D)}\right] \tag{2.11}$$

and the RSC encoding process can be described by

$$r_i = m_i + \sum_{j=1}^{M_c} r_{i-j} g_j^{(0)} \qquad i \in \{0, \ldots, N_b - 1\} \tag{2.12}$$

where r_i is the feedback variable, and the parity outputs $x_i^{(q)}$ are given by

$$x_i^{(q)} = \sum_{j=0}^{M_c} r_{i-j} g_j^{(q)} \qquad i \in \{0, \ldots, N_b - 1\} \tag{2.13}$$

2.3.2 Interleaving

Typically, the output codewords of a RSC encoder have a high Hamming weight[5]. However, some input sequences can cause the RSC encoder to produce low weight[6] codewords. The combination of interleaving (permuting) and RSC encoding ensures that the codewords produced by a turbo encoder have a high Hamming weight [40]. For instance, assume that the RSC encoder C_1 receives an input sequence which causes it to generate a low weight output. Then, it is improbable that the other RSC encoder, C_2,—which receives the interleaved version of the input—will also produce a low weight output. Hence, the interleaver spreads the low weight input sequences, so that the resulting codewords have a high Hamming weight [28], [41]. This interleaving operation can be defined as

$$\bar{m}_{\pi(i)} = m_i \qquad i \in \{0, \ldots, N_b - 1\} \tag{2.14}$$

where m_i and $\bar{m}_{\pi(i)}$ are the i-th input and the output of the interleaver, N_b is the size of the interleaver, and π is some interleaving function. The amount of spreading or the efficiency of the interleaver depends on its size, and on the type of the interleaving function used. Moreover, it is this interleaving operation that gives the turbo code its random-like characteristic.

Several different types of interleavers have been used with turbo codes. These include non-random or block, semi-random, and pseudo-random interleavers. Block interleavers are defined by a matrix with v_r rows and v_c columns, such that $N_b = v_r \times v_c$. The interleaver works by writing data row-wise (or column-wise) into the matrix, and by reading data column-wise (or row-wise) out of the matrix. Block interleavers are effective if the low weight input sequence is confined to a row (assuming that data is read out of the matrix in a column-wise manner). If the low weight input sequences are confined to several consecutive rows, then the v_c columns of the interleaver need to be read out in a manner specified in [42]. However, the interleaver may still fail to spread certain sequences. In such cases, the method proposed in [42] can be used to first permute the columns and then the rows.

Semi-random or S-random interleavers work by generating random integers i, $1 \leq i \leq N_b$, without replacement [41], [43]. Each randomly selected integer is compared to S previously selected integers. If the current selection is equal to any S previous selections within a distance of $\pm S$, then the current selection is rejected. This process is repeated until all the N_b integers have been selected. Typically, S is chosen to be less than $\sqrt{N_b/2}$ in order to minimize the search time. However, a given instance of the algorithm is not guaranteed to finish the search process successfully.

Pseudo-random interleavers map a bit position i to some other location j, $j \leq N_b$, according

[5]The Hamming weight of a codeword is the Hamming distance (number of positions at which two codewords differ) between the codeword and the all-zero codeword.

[6]The smallest Hamming weight of all the valid codewords is known as the minimum distance of the code. The error detection and correction capabilities of a code are directly proportional to its minimum distance.

to a prescribed, but randomly (pseudo-randomly) generated rule. Pseudo-random interleavers possess little structure, and have, therefore, been shown to perform better than both the block and the S-random interleavers. In general, for a turbo code with two RSC codes separated by a pseudo-random interleaver, the probability that a weight w input sequence will be reproduced somewhere within the turbo encoder is given by [41]

$$1 - \left[1 - \frac{\beta}{N_b^{w-1}}\right]^{N_b} \tag{2.15}$$

where β is a constant that depends on the weight w. It is evident from (2.15) that the probability of the aforementioned event decreases rapidly as the size of the pseudo-random interleaver increases.

2.3.3 Trellis Termination

An important issue that affects the performance of turbo codes is trellis[7] termination. Trellis termination helps impose a known boundary condition upon the trellis of a RSC code, which in turn aids in the decoding of turbo codes[8]. While it is possible to terminate the trellis of a constituent RSC encoder with a tail of M_c bits, the simultaneous termination of both trellises is a non trivial task. There are two complicating factors. First, due to interleaving, the tail bits for the second (bottom) RSC encoder may not be at the end of its input stream. Second, the process of interleaving and the recursive nature of RSC encoding complicate the computation of the tail bits suitable for both RSC encoders. Several solutions have been proposed to this trellis termination problem. For instance, the simplest solution requires that only the trellis associated with the top RSC encoder be terminated [40]. Another solution is based on a circular-shift interleaver that uses a single tail of M_c bits to terminate both trellises [40]. However, circular-shift registers add an extra level of structure to the interleaver, and, therefore, degrade the performance of the turbo encoder.

A more viable scheme for trellis termination uses non-interleaved tail bits for trellis termination [44]. The input sequence, $\mathbf{m}$, is fed into the first RSC encoder C_1. A short sequence, $\mathbf{m}_c$, is constructed from the state of C_1 after the input sequence has been completely encoded. The RSC encoder C_1 is then forced back to the all-zero state by using the sequence $\mathbf{m}_c$ as its new input. The original input sequence is interleaved and fed into the second RSC encoder C_2. The second RSC encoder is likewise reinitialized to zero by the sequence $\bar{\mathbf{m}}_c$. Hence, the transmitted sequence contains the original N_b bit information sequence, and the two $N_b + M_c$ bit parity sequences. This approach suffers from the fact that the inputs of the two RSC encoders are different. This reduces the efficiency of the decoding algorithm, which is derived under the assumption that the two encoders receive the same set of input data.

[7] A trellis is the time expanded state diagram of a code.

[8] Section(2.6.1) describes how the turbo decoder uses these boundary conditions to initialize the trellis for decoding each constituent RSC code.

A fourth scheme uses precursor bits in a two-pass process to terminate the two trellises [45], [46]. During the first pass, the first $2M_c$ bits out of the N_b total bits of the input sequence are set to zero. These $2M_c$ bits are called the precursor bits. The input sequence is passed through the first RSC encoder, interleaved, and then passed through the second RSC encoder. At the end of the encoding process, the final states of the two RSC encoders are examined. These two states are used as indices in a look-up table (LUT) to determine the precursor bits necessary to terminate the two trellises. The precursor bits are set to the value chosen from the LUT, and a second pass of encoding is performed. This second pass ensures that both trellises terminate in the all-zero state. This approach requires that there exists a one-to-one correspondence between the final states of the two RSC encoders and the values of the precursor bits. Hence, this approach places a restriction on the design of the interleaver. However, this restriction is less severe, and does not degrade the performance significantly. Furthermore, in this scheme the entire input sequence (including the precursor bits) is interleaved. This eliminates the bias (and yields better performance) inherent in schemes that use non-interleaved tail bits for trellis termination.

2.4 Theoretical Performance Analysis

Turbo codes perform well at low signal-to-noise ratios because the number of low weight codewords is small. However, the performance of turbo codes is limited at high signal-to-noise ratios by their relatively small minimum distance (due to simple constituent codes). This effect of weight and minimum distance on the performance of turbo codes can be quantified in terms of the conditional weight enumerating function (CWEF). For a (n, k) code, the CWEF is defined as [47], [49]

$$A(w, H) = \sum_{h=0}^{n} A_{w,h} H^h \tag{2.16}$$

where $A_{w,h}$ is the number of codewords with Hamming weight h that are generated by an input of weight w. The CWEF, therefore, describes the weight distribution of all the codewords generated by an input of a certain weight.

To facilitate our theoretical analysis, we borrow the concept of a uniform interleaver from [48]. The uniform interleaver (UI) is a probabilistic device that represents the average behaviour of all deterministic interleavers of size N_b. The UI maps a given input sequence of length N_b and weight w into all distinct $\begin{pmatrix} N_b \\ w \end{pmatrix}$ permutations with equal probability $1/\begin{pmatrix} N_b \\ w \end{pmatrix}$. The UI, therefore, makes independent the weight distributions of the parity check bits generated by the first and the second RSC encoder. Moreover, it permits the estimation of an overall interleaver gain, independent of the particular interleaver used, and decouples the roles played by the interleaver and the RSC encoders in determining the overall performance of

the turbo code. Therefore, the overall CWEF for a parallel concatenation of two rate 1/2 RSC codes, C_1 and C_2, is given by [47], [49]

$$A_{w,h}^{C_p} = \sum_{h_1} \frac{A_{w,h_1}^{C_1} \times A_{w,h-h_1}^{C_2}}{\begin{pmatrix} N_b \\ w \end{pmatrix}} \tag{2.17}$$

Note that the codeword weight of C_2 does not include the weight of the systematic bits since they are not transmitted. An upper bound on the bit error probability can be computed by using the CWEF, and by assuming maximum-likelihood[9] (ML) decoding over an additive, white, Gaussian noise (AWGN) channel

$$P_b(e) \leq \sum_h \left(\sum_w \frac{w}{N_b} A_{w,h}^{C_p} \right) Q\left[\sqrt{2hr \frac{E_b}{N_o}} \right] \tag{2.18}$$

where E_b is the energy per bit, N_o is the single-sided noise power spectral density, r is the code rate, and the Q-function is defined as

$$Q[x] = \frac{1}{\sqrt{2\pi}} \int_x^\infty e^{-x^2/2} dx \tag{2.19}$$

Consider the transmission of an all-zero codeword. An error event occurs whenever a path diverges from the all-zero state at some time instant, and then re-merges with the all-zero state at some other time instant. Let A_{w,h,n_i} be the number of codewords with input weight w, output weight h, and n_i concatenated error events, where $i \in \{1, 2\}$ is the index of the two RSC codes. If the interleaver size N_b is much larger than the constraint length of the RSC codes, then $A_{w,h}$ can be approximated as [49]

$$A_{w,h} \approx \sum_{j=1}^{n_m} \begin{pmatrix} N_b \\ j \end{pmatrix} A_{w,h,j} \tag{2.20}$$

where n_m is the largest number of error events concatenated in a codeword of weight h generated by an input of weight w. Using (2.17) and (2.20), the overall CWEF can be written as

$$A_{w,h}^{C_p} = \sum_{n_1=1}^{n_{1m}} \sum_{n_2=1}^{n_{2m}} \frac{\begin{pmatrix} N_b \\ n_1 \end{pmatrix} \begin{pmatrix} N_b \\ n_2 \end{pmatrix}}{\begin{pmatrix} N_b \\ w \end{pmatrix}} \sum_{h_1} A_{w,h_1,n_1}^{C_1} \times A_{w,h-h_1,n_2}^{C_2} \tag{2.21}$$

[9]The sub-optimal iterative decoding algorithm used to decode turbo codes offers near ML performance. However, its analysis is cumbersome. We, therefore, assume a ML decoder to estimate the bit error probability.

If $N_b >> n$, then we can use the approximation [49]

$$\left(\begin{array}{c} N_b \\ n \end{array} \right) \approx \frac{N_b^n}{n!} \tag{2.22}$$

Hence, (2.21) can be written as

$$A_{w,h}^{C_p} = \sum_{n_1=1}^{n_{1m}} \sum_{n_2=1}^{n_{2m}} \frac{w!}{n_1! n_2!} N_b^{n_1+n_2-w} \sum_{h_1} A_{w,h_1,n_1}^{C_1} \times A_{w,h-h_1,n_2}^{C_2} \tag{2.23}$$

and the bit error probability in (2.18) can be rewritten as

$$P_b(e) \leq \sum_h \left(\sum_w w w! \sum_{n_1=1}^{n_{1m}} \sum_{n_2=1}^{n_{2m}} \frac{w!}{n_1! n_2!} N_b^{n_1+n_2-w-1} \sum_{h_1} A_{w,h_1,n_1}^{C_1} \times A_{w,h-h_1,n_2}^{C_2} \right) Q\left[\sqrt{2hr \frac{E_b}{N_o}} \right] \tag{2.24}$$

It is evident that for each h_1, as $N_b \to \infty$, the dominant term in (2.24) is the one with the largest exponent of N_b. Define this largest exponent as

$$\alpha(h) = \max_{n_1,n_2,w} \{n_1(h) + n_2(h) - w(h) - 1\} \tag{2.25}$$

where $n_1(h)$ and $n_2(h)$ are the number of error events concatenated in a weight h codeword generated by a weight w input to RSC encoders C_1 and C_2. Therefore, the bit error probability can be approximated as

$$P_b(e) \approx \sum_h C_h N_b^{\alpha(h)} Q\left[\sqrt{2hr \frac{E_b}{N_o}} \right] \tag{2.26}$$

where C_h is a constant independent of N_b, and is a measure of the multiplicity of weight h codewords. Therefore, the bit error probability improves as the multiplicity of low weight codewords decreases. Furthermore, if $\alpha(h) < 0$, the bit error probability decreases as N_b increases. This effect is called the interleaving gain. Is is shown in [47] and [49] that $\alpha(h) \leq -1$ for a RSC code, and that it is always greater than zero for a non-recursive convolutional code. The interleaving gain is, therefore, only realizable for RSC codes. At high signal-to-noise ratios, the bit error probability can be further approximated by the minimum distance asymptote

$$P_b(e) \approx C_{h_{min}} N_b^{\alpha(h_{min})} Q\left[\sqrt{2h_{min} r \frac{E_b}{N_o}} \right] \tag{2.27}$$

where h_{min} is the minimum distance of the turbo code. The minimum distance asymptote gives rise to an error floor, which is a low-slope region where the bit error rate decreases slowly with increasing signal-to-noise ratio. The error floor can be lowered by increasing the interleaver size. However, the slope of the error floor is limited by the magnitude of h_{min}, which is typically small for turbo codes because of the simplicity of the constituent RSC codes.

2.5 Iterative Turbo Decoding

The problem of turbo decoding is tantamount to estimating the Markov process associated with each of the two constituent RSC codes. Although turbo codes can be modeled as a single Markov process, the presence of an interleaver complicates the structure of the resulting trellis, and yields exorbitantly complex decoding algorithms. Therefore, the two Markov processes are estimated independently using two separate decoders (henceforth referred to as constituent decoders). However, the two constituent decoders operate in a symbiotic manner, i.e., the output of one feeds the input of the other, and *vice versa*. This symbiotic relationship is realized by the fact that the two RSC encoders (the two Markov processes) are driven by the same set of input data. Turbo decoding, therefore, proceeds in an iterative manner, where each iteration corresponds to the aforementioned symbiotic cycle.

For a rate $1/n$ RSC code, a constituent decoder accepts the received systematic symbol $y_i^{(0)}$, the received parity symbols $\{y_i^{(1)}, \ldots, y_i^{(n-1)}\}$, and the soft output of the second decoder. It computes the log-likelihood ratio[10] (LLR)

$$\Lambda_i = \ln \left[\frac{Pr\{m_i = 1|\mathbf{y}\}}{Pr\{m_i = 0|\mathbf{y}\}} \right] \tag{2.28}$$

where m_i is the systematic output of the RSC enoder (i.e., $m_i = x_i^{(0)}$), $\mathbf{y}$ is the received sequence, and $Pr\{m_i = b|\mathbf{y}\}$, $b \in \{0, 1\}$, is the *a posteriori* probability (APP) of the i-th message symbol. Since the constituent decoders compute the AP probability and operate on soft information, they are commonly referred to as soft-in soft-out (SISO) APP decoders.

Using Baye's rule, (2.28) can be rewritten as

$$\Lambda_i = \ln \left[\frac{Pr\{\mathbf{y}|m_i = 1\}}{Pr\{\mathbf{y}|m_i = 0\}} \right] + \ln \left[\frac{Pr\{m_i = 1\}}{Pr\{m_i = 0\}} \right] \tag{2.29}$$

where the second term represents the *a priori* information about the message symbol m_i. For a rate 1/3 turbo code (two constituent decoders), this *a priori* information can be expressed

[10]The LLR is synonymous with soft information, and is a measure of reliability.

as the sum

$$\ln\left[\frac{Pr\{m_i=1\}}{Pr\{m_i=0\}}\right] = W_i + Z_i \tag{2.30}$$

where W_i is the *extrinsic* (new information derived during the current decoding iteration) information extracted from the output of the constituent decoder, and Z_i is the soft information derived from the second decoder.

Consider a communication system with binary PSK modulation, coherent detection, and a channel model

$$y_i = a_i(2x_i - 1) + n_i \tag{2.31}$$

where $x_i \in \{0, 1\}$ is the transmitted symbol, y_i is the received symbol, a_i is the channel gain, n_i is a zero-mean Gaussian noise sample with variance $N_o/2E_s$, E_s[11] is the energy per symbol, and $N_o/2$ is the double-sided noise power spectral density. Using the aforementioned channel model, (2.29) can be rewritten as [50]

$$\Lambda_i = 4a_i^{(0)}\frac{E_s}{N_o}y_i^{(0)} + W_i + Z_i \tag{2.32}$$

and the extrinsic information derived from the output of the constituent decoder can be expressed as

$$W_i = \Lambda_i - 4a_i^{(0)}\frac{E_s}{N_o}y_i^{(0)} - Z_i \tag{2.33}$$

Figure (2.5) shows the block diagram for a rate 1/3 turbo decoder. The first constituent decoder, D_1, receives the scaled systematic received sequence $\mathbf{y}^{(0)}$, the first scaled parity received sequence $\mathbf{y}^{(1)}$, and the *a priori* information Z_2 from the second decoder D_2. It computes the LLRs Λ_1. The extrinsic information W_1 is extracted from Λ_1 using (2.33). This extrinsic information is interleaved (the interleaver used in the turbo decoder is same as the interleaver used in the turbo encoder), and is used as *a priori* information Z_1 by the second decoder. The second decoder, D_2, also receives the interleaved and scaled systematic received sequence $\bar{\mathbf{y}}^{(0)}$, and the second scaled parity sequence $y^{(2)}$. It generates the LLRs Λ_2. The extrinsic information W_2 is extracted from Λ_2 using (2.33), is de-interleaved, and used as *a priori* information by D_1 during the next decoding iteration. After η iterations, the final estimate of the message sequence $\hat{\mathbf{m}}$ is found by de-interleaving and hard-limiting the output of D_2, i.e.,

$$\hat{m}_i = \begin{cases} 1 & \Lambda_{2_{\pi^{-1}(i)}} \geq 0 \\ 0 & \Lambda_{2_{\pi^{-1}(i)}} < 0 \end{cases} \tag{2.34}$$

[11] $E_s = rE_b$, where r is the code rate, and E_b is the energy per bit.

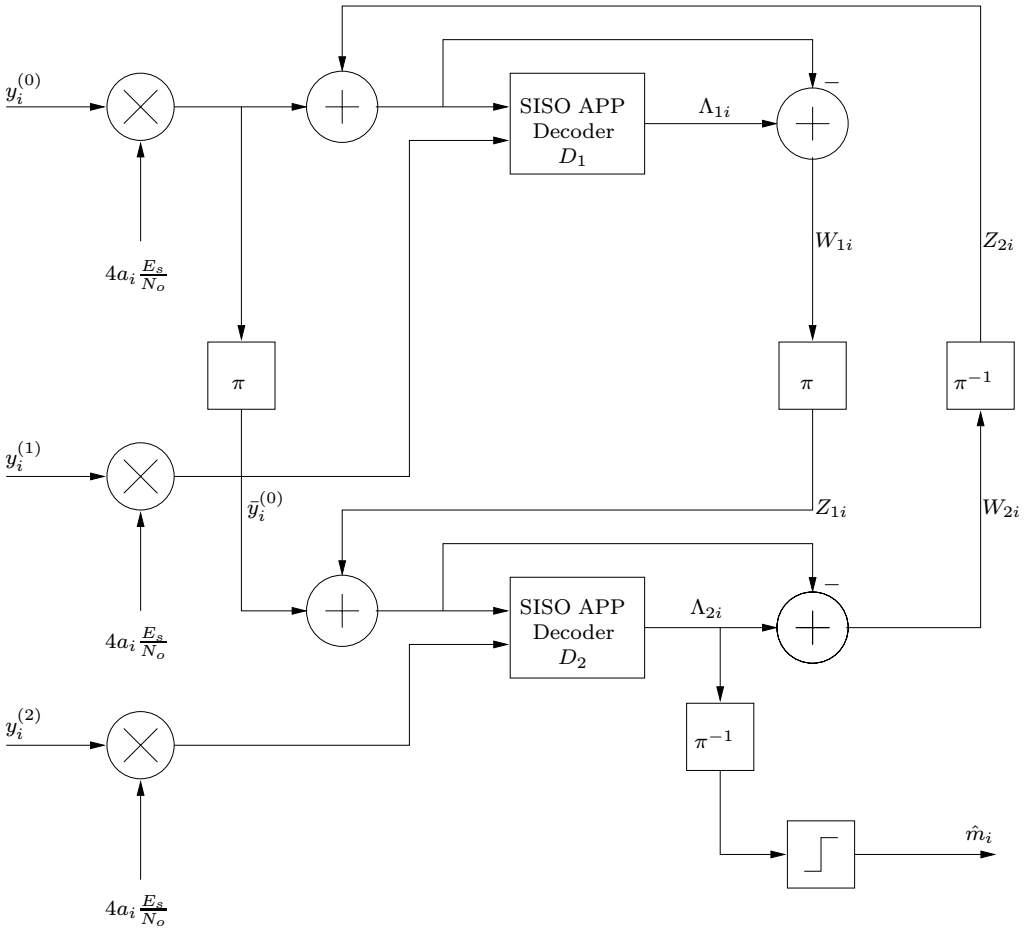

Figure 2.3: A rate 1/3 turbo decoder.

2.6 A Posteriori Probability Algorithms

Several different decoding strategies have been adopted for the SISO APP decoding of turbo codes. These decoding schemes can be classified as either maximum likelihood sequence estimation (MLSE) algorithms, or as symbol-by-symbol maximum *a posteriori* (MAP) algorithms. As their names suggest, the two aforementioned algorithms differ in the way they estimate a given Markov process. For instance, MLSE algorithms attempt to estimate the most probable state sequence, $\hat{\mathbf{s}}$, of the Markov process, given a received sequence $\mathbf{y}$

$$\hat{\mathbf{s}}_{MLSE} = \arg\max_{\mathbf{s}}[Pr\{\mathbf{s}|\mathbf{y}\}] \tag{2.35}$$

Contrariwise, MAP algorithms find the most likely individual states, $\hat{s}_i$, given $\mathbf{y}$

$$\hat{\mathbf{s}}_{MAP} = \arg\max_{s_i}[Pr\{\mathbf{s}|\mathbf{y}\}] \tag{2.36}$$

Hence, the sequence $\hat{\mathbf{s}}_{MLSE}$ traverses a continuous path through the trellis defined by the Markov process. However, the sequence $\hat{\mathbf{s}}_{MAP}$ need not form such a continuous path. Therefore, MLSE algorithms minimize the codeword error probability or the state estimation error rate (SER), while MAP algorithms minimize the symbol error probability or bit error rate (BER).

The MLSE family of APP decoding algorithms include the soft output Viterbi algorithm (SOVA) [51], and the Improved SOVA algorithm [52]. The MAP family of decoding algorithms include the modified BCJR[12] algorithm [53]—originally used by Berrou et al. It also includes the log-MAP and the Max-Log-MAP algorithms [54], [55]. For this thesis, we shall focus only on the MAP decoding algorithms.

2.6.1 Modified BCJR Algorithm

The symbol-by-symbol MAP algorithm—also known as the BCJR algorithm—was developed by Bahl et al as an alternative to the Viterbi algorithm [56] for decoding convolutional codes [53]. When applied to turbo codes, a modified BCJR is used to account for the recursive nature of iterative turbo decoding. The BCJR algorithm computes the APP of the message symbols, and the LLR

$$\Lambda_i = \ln\left[\frac{Pr\{m_i = 1|\mathbf{y}\}}{Pr\{m_i = 0|\mathbf{y}\}}\right] \tag{2.37}$$

where m_i is a message symbol (systematic output of the RSC enoder), and $\mathbf{y}$ is the received sequence of length N_b. By incorporating the RSC code's trellis, (2.37) can be rewritten as

$$\Lambda_i = \ln\left[\frac{\sum_{S^+} Pr\{s_{i-1} = s^{'}, s_i = s, \mathbf{y}\}/Pr\{\mathbf{y}\}}{\sum_{S^-} Pr\{s_{i-1} = s^{'}, s_i = s, \mathbf{y}\}/Pr\{\mathbf{y}\}}\right] \tag{2.38}$$

where $s_i \in S = \{0, \ldots, 2^{M_c} - 1\}$ is the state of the encoder at time i, S^+ is the set of ordered pairs $(s^{'}, s)$ corresponding to all state transitions $(s_{i-1} = s^{'}) \rightarrow (s_i = s)$ caused by the message symbol $m_i = 1$, and S^- is likewise defined for $m_i = 0$. The probability $Pr\{s_{i-1} = s^{'}, s_i = s, \mathbf{y}\}$ can be expressed as the product of three probabilities

$$Pr\{s_{i-1} = s^{'}, s_i = s, \mathbf{y}\} = \alpha_{i-1}(s^{'})\gamma_i(s^{'}, s)\beta_i(s) \tag{2.39}$$

where the α, β, and γ probabilities are defined as

$$\begin{aligned} \alpha_i(s) &= Pr\{s_i = s, \mathbf{y}_i^i\} \\ \beta_i(s) &= Pr\{\mathbf{y}_{i+1}^{N_b}|s_i = s\} \\ \gamma_i(s^{'}, s) &= Pr\{s_i = s, y_i|s_{i-1} = s^{'}\} \end{aligned} \tag{2.40}$$

[12]The BCJR algorithm is named in honor of its inventors L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv.

Hence, the α probability represents the probability of being in state s, given the partial received sequence until time i. Likewise, the β probability represents the probability of being in state s, given the partial received sequence after time i. The γ probability represents the probability of transition from state $s^{'}$ to state s at time i.

The α probabilities are computed via a forward recursion through the trellis

$$\alpha_i(s) = \sum_{s^{'}} \alpha_{i-1}(s^{'})\gamma_i(s^{'}, s) \qquad i \in \{1, 2, \ldots, N_b - 1\} \tag{2.41}$$

For large N_b, the α probabilities approach zero geometrically fast, and must, therefore, be normalized at each time index to maintain accuracy and stability. Hence,

$$\alpha_i(s) = \frac{\sum_{s^{'}} \alpha_{i-1}(s^{'})\gamma_i(s^{'}, s)}{\sum_s \sum_{s^{'}} \alpha_{i-1}(s^{'})\gamma_i(s^{'}, s)} \qquad i \in \{1, 2, \ldots, N_b - 1\} \tag{2.42}$$

Furthermore, since the RSC encoder starts in the all-zero state, the α probabilities are initialized as

$$\alpha_0(s) = \begin{cases} 1 & s = 0 \\ 0 & s \neq 0 \end{cases} \tag{2.43}$$

The β probabilities are computed and normalized via a backward recursion through the trellis

$$\beta_{i-1}(s^{'}) = \frac{\sum_s \beta_i(s)\gamma_i(s^{'}, s)}{\sum_s \sum_{s^{'}} \alpha_{i-1}(s^{'})\gamma_i(s^{'}, s)} \qquad i \in \{N_b - 2, N_b - 1, \ldots, 0\} \tag{2.44}$$

If the RSC encoder terminates in the all-zero state, then the β probabilities are initialized as

$$\beta_{N_b-1}(s) = \begin{cases} 1 & s = 0 \\ 0 & s \neq 0 \end{cases} \tag{2.45}$$

Contrariwise, if the RSC encoder is left unterminated, then

$$\beta_{N_b-1}(s) = \frac{1}{2^{M_c}} \qquad \forall\ s \tag{2.46}$$

The γ probability defined in (2.40) can be expressed as

$$\gamma_i(s^{'}, s) = Pr\{m_i\}Pr\{y_i|x_i\} \tag{2.47}$$

where x_i and y_i are the transmitted and the received codewords associated with the message symbol m_i. The probability $Pr\{m_i\}$ is obtained from the *a priori* information Z_i at the output of the second APP decoder

$$Pr\{m_i\} = \begin{cases} \frac{e^{Z_i}}{1+e^{Z_i}} & m_i = 1 \\ \frac{1}{1+e^{Z_i}} & m_i = 0 \end{cases} \tag{2.48}$$

Alternately, the probability $Pr\{m_i\}$ can be written in a more compact form as

$$Pr\{m_i\} = \frac{\left(e^{Z_i}\right)^{m_i}}{1+e^{Z_i}} \tag{2.49}$$

Using Baye's theorem, the *a priori* probability $Pr\{y_i|x_i\}$ in (2.47) can be rewritten as

$$Pr\{y_i|x_i\} = \frac{Pr\{x_i|y_i\}Pr\{x_i\}}{Pr\{y_i\}} = Pr\{x_i|y_i\}C \tag{2.50}$$

where C is constant for a given codeword, and can, therefore, be ignored from all future computations . Define the channel reliability variable, $R_i^{(q)}$, as

$$R_i^{(q)} = \ln\left[\frac{Pr\{x_i^{(q)} = 1|\mathbf{y}\}}{Pr\{x_i^{(q)} = 0|\mathbf{y}\}}\right] \qquad q \in \{0, 1, \ldots, n-1\} \tag{2.51}$$

Using Baye's theorem the AP probability $Pr\{x_i^{(q)} = b|\mathbf{y}\}$, $b \in \{0, 1\}$, can be written in terms of the *a priori* probability $Pr\{y_i^{(q)}|x_i^{(q)} = b\}$. Furthermore, using the channel model defined in (2.31), the probability $Pr\{y_i^{(q)}|x_i^{(q)}\}$ can be expressed as

$$Pr\{y_i|x_i\} = \frac{1}{\sqrt{\pi N_o/E_s}} \exp\left\{\frac{-E_s}{N_o}\left[y_i^{(q)} - a_i^{(q)}(2x_i^{(q)} - 1)\right]^2\right\} \tag{2.52}$$

Consequently the channel reliability variable can be simplified as

$$R_i^{(q)} = 4a_i^{(q)}\frac{E_s}{N_o}y_i^{(q)} \tag{2.53}$$

and the AP probabilities $Pr\{x_i^{(q)} = b|y_i^{(q)}\}$, $b \in \{0, 1\}$, can either be written as

$$Pr\{x_i^{(q)}|y_i^{(q)}\} = \begin{cases} \frac{e^{R_i^{(q)}}}{1+e^{R_i^{(q)}}} & x_i^{(q)} = 1 \\ \frac{1}{1+e^{R_i^{(q)}}} & x_i^{(q)} = 0 \end{cases} \tag{2.54}$$

or alternately as

$$Pr\{x_i^{(q)}|y_i^{(q)}\} = \frac{\left(e^{R_i}\right)^{x_i^{(q)}}}{1+e^{R_i^{(q)}}} \tag{2.55}$$

Therefore, the γ probability in (2.47) can be rewritten as

$$\gamma_i(s^{'}, s) = \frac{\left(e^{Z_i}\right)^{m_i}}{1+e^{Z_i}} \prod_{q=0}^{n-1} \frac{\left(e^{R_i^{(q)}}\right)^{x_i^{(q)}}}{1+e^{R_i^{(q)}}} \tag{2.56}$$

Since the denominator in (2.56) is constant for a given codeword, the computation of the γ probability can be simplified as

$$\gamma_i(s^{'}, s) = \left(e^{Z_i}\right)^{m_i} \prod_{q=0}^{n-1} \left(e^{R_i^{(q)}}\right)^{x_i^{(q)}} \tag{2.57}$$

By using the α, β, and γ probabilities, the BCJR algorithm computes the LLR

$$\Lambda_i = \ln\left[\frac{\sum_{S^+} Pr\{\alpha_{i-1}(s^{'})\gamma_i(s^{'}, s)\beta_i(s)\}}{\sum_{S^-} Pr\{\alpha_{i-1}(s^{'})\gamma_i(s^{'}, s)\beta_i(s)\}}\right] \tag{2.58}$$

Note that the term $Pr\{\mathbf{y}\}$ in (2.38) cancels out, and, therefore, need not be computed. Furthermore, the extrinsic information, W, can be computed using (2.33).

2.6.2 Log-MAP and the Max-Log-MAP Algorithms

Although optimal, the BCJR algorithm offers several technical difficulties. These include sensitivity to numeric representation, the necessity of non-linear functions, and a high number of additions and multiplications. These technical difficulties can be alleviated by implementing sub-optimal algorithms such as the Log-MAP algorithm or the Max-Log-MAP algorithm [54], [55]. As their names suggest, these algorithms compute the α, β, and the γ probabilities in the log-domain. Both algorithms perform multiplications in the log-domain as additions. However, they perform additions in the log-domain by using an approximation of the Jacobian logarithm[13]. This approximation is known as the $\max^*(\cdot)$ function, and is defined as [54]

$$\max^*(x, y) = \begin{cases} \max(x, y) & \text{Max-Log-MAP algorithm} \\ \max(x, y) + f_c(|y - x|) & \text{Log-MAP algorithm} \end{cases} \tag{2.59}$$

[13]The Jacobian logarithm describes the addition of two exponentials in the log-domain as $\ln[e^x + e^y] = \max(x, y) + \ln(1 + e^{-|y-x|})$.

where $f_c(\cdot)$ is the associated correction function. Therefore, the only difference between the Log-MAP algorithm and the Max-Log-MAP algorithm lies in the implementation of the $\max^*(\cdot)$ function[14].

Let $\bar{\alpha}$, $\bar{\beta}$, and $\bar{\gamma}$ represent the natural logarithm of the α, β, and the γ probabilities. Hence, $\bar{\gamma}$ can be expressed as

$$\bar{\gamma}_i(s^{'}, s) = m_i Z_i + \sum_{q=0}^{n-1} x_i^{(q)} R_i^{(q)} \tag{2.60}$$

or alternately as

$$\bar{\gamma}_i(s^{'}, s) = \begin{cases} \sum_{q=1}^{n-1} x_i^{(q)} R_i^{(q)} & m_i = x_i^{(0)} = 0 \\ Z_i + R_i^{(0)} + \sum_{q=1}^{n-1} x_i^{(q)} R_i^{(q)} & m_i = x_i^{(0)} = 1 \end{cases} \tag{2.61}$$

Likewise, $\bar{\alpha}$ and $\bar{\beta}$ can be computed and normalized as

$$\bar{\alpha}_i(s) = \ln \left\{ \sum_{s'} \exp \left[\bar{\alpha}_{i-1}(s^{'}) + \bar{\gamma}_i(s^{'}, s) \right] \right\} - \max_s \left[\bar{\alpha}_i(s) \right] \tag{2.62}$$

$$= \max_{s'}{}^* \left[\bar{\alpha}_{i-1}(s^{'}) + \bar{\gamma}_i(s^{'}, s) \right] - \max_s \left[\bar{\alpha}_i(s) \right] \tag{2.63}$$

and

$$\bar{\beta}_{i-1}(s^{'}) = \ln \left\{ \sum_{s} \exp \left[\bar{\beta}_i(s) + \bar{\gamma}_i(s^{'}, s) \right] \right\} - \max_{s'} \left[\bar{\beta}_{i-1}(s^{'}) \right] \tag{2.64}$$

$$= \max_{s}{}^* \left[\bar{\beta}_i(s) + \bar{\gamma}_i(s^{'}, s) \right] - \max_{s'} \left[\bar{\beta}_{i-1}(s^{'}) \right] \tag{2.65}$$

Since the RSC encoder starts in the all-zero state, $\bar{\alpha}$ is initialized as

$$\bar{\alpha}_0(s) = \begin{cases} 0 & s = 0 \\ -\infty & s \neq 0 \end{cases} \tag{2.66}$$

If the RSC encoder terminates in the all-zero state, then $\bar{\beta}$ is initialized as

$$\bar{\beta}_{N_b-1}(s) = \begin{cases} 0 & s = 0 \\ -\infty & s \neq 0 \end{cases} \tag{2.67}$$

[14]The absence of the correction function from the Max-Log-MAP algorithm causes it to perform 0.5 dB worse that the Log-MAP algorithm at low signal-to-noise ratios.

Contrariwise, if the RSC encoder is left unterminated, then

$$\bar{\beta}_{N_b-1}(s) = 0 \qquad \forall\ s \tag{2.68}$$

Once $\bar{\alpha}$, $\bar{\beta}$, and $\bar{\gamma}$ are known, the LLR can be found using

$$\begin{aligned}\Lambda_i = &\max_{S^+}{}^{*} \left[\bar{\alpha}_{i-1}(s^{'}) + Z_i + R_i^{(0)} + \sum_{q=1}^{n-1} x_i^{(q)} R_i^{(q)} + \bar{\beta}_i(s))\right] \\ &- \max_{S^-}{}^{*} \left[\bar{\alpha}_{i-1}(s^{'}) + \sum_{q=1}^{n-1} x_i^{(q)} R_i^{(q)} + \bar{\beta}_i(s))\right]\end{aligned} \tag{2.69}$$

Equation (2.69) can be further simplified as

$$\begin{aligned}\Lambda_i = Z_i + R_i^{(0)} &+ \max_{S^+}{}^{*} \left[\bar{\alpha}_{i-1}(s^{'}) + \sum_{q=1}^{n-1} x_i^{(q)} R_i^{(q)} + \bar{\beta}_i(s))\right] \\ &- \max_{S^-}{}^{*} \left[\bar{\alpha}_{i-1}(s^{'}) + \sum_{q=1}^{n-1} x_i^{(q)} R_i^{(q)} + \bar{\beta}_i(s))\right]\end{aligned} \tag{2.70}$$

Note that the first term in (2.70) represents the *a priori* information from the second APP decoder, the second term represents the channel reliability for the systematic information bits, and the third and fourth terms represent the extrinsic information.

2.7 Chapter Summary

Perhaps the most significant invention in the field of information theory since its inception in the 1950s was the discovery of turbo codes in 1993. The remarkable performance of turbo codes, i.e, their ability to perform extremely close to the Shannon bound, was subject to early skepticism. However, the last few years have seen a growing interest in the use of turbo codes for error correction, and more so the use of *turbo processing (decoding)* for a wide variety of applications in communications.

This chapter presented an overview of the turbo codec. The chapter described in detail the turbo encoding scheme. It described the different constituent components that make up a turbo encoder, and presented a theoretical framework for analyzing their effect on the performance of turbo codes. It was shown that the design of *good* turbo codes entail the use of recursive systematic convolutional (RSC) codes with large minimum free distance and small multiplicity of low weight codewords, the use of large psuedo-random interleavers, and the need for trellis termination for the constituent RSC encoders. The chapter also described the

turbo or iterative decoding principle. It highlighted the need for the independent decoding of the constituent RSC codes, and the symbiotic exchange of soft information between the corresponding decoders. The chapter presented the optimal maximum *a posteriori* MAP (BCJR) algorithm, and the sub-optimal (but more computationally efficient) Log-MAP and the Max-Log-MAP algorithms for decoding the constituent RSC codes.

Chapter 3

Markov Fading Channels

Devised nearly a hundred years ago to answer a mathematical curiosity[1], Markov models have seen widespread use in fields as diverse as engineering, economics, and genetics. Apropos engineering, Markov models have been widely used in speech processing, pattern recognition, artificial intelligence, queuing theory, and communications theory. For instance, Markov models have been used since the 1960s in the analysis and simulation of digital communication systems [57], [58], [59], [60]. Of special significance, has been the use of Markov models in the study of wireless communication systems. Typical applications have included signal/sequence estimation [61], [62], [63], frequency tracking [64], equalization [65], and burst error simulation [66], [67] ,[68], [69]. However, more pertinent to this thesis has been the recent interest in the use of Markov models to characterize fading channels [70], [71], [72].

This chapter describes the application of Markov models in characterizing fading channels. Sections (3.1) and (3.2) present an introduction to finite state channels and hidden Markov models. Section (3.3) describes a Markov fading channel. Section (3.4) describes the different techniques used to estimate Markov models for fading channels. Section (3.5) presents the simulation results used to quantify the accuracy of the different estimation techniques described in the previous section.

3.1 Finite State Channels

A discrete finite state channel (FSC) model is one in which the channel can occupy any state from the state space $S = \{1, 2, \ldots, u\}$. If at time $t-1$ the channel is in state $s_{t-1} \in S$, and if at time t the input to the channel is $a_t \in A$, then the channel outputs the symbol $b_t \in B$

[1] In 1907, Andrei Andreivich Markov formulated the concept of Markov chains—in his seminal paper "Extension of the limit theorems of probability theory to a sum of variables connected in a chain"—while investigating the applicability of the law of large numbers to dependent and independent random variables.

and transfers to state $s_t \in S$ with probability $Pr\{b_t, s_t|a_t, s_{t-1}\}$. The joint probability of the final state s_t and the received sequence $\mathbf{b_1^t} = (b_1, b_2, \ldots, b_t)$ conditioned upon the initial state s_0 and the transmitted sequence $\mathbf{a_1^t}$ is given by [60]

$$Pr\{\mathbf{b_1^t}, s_t|\mathbf{a_1^t}, s_0\} = \sum_{s_1^{t-1}} \prod_{i=1}^{t} Pr\{b_i, s_i|a_i, s_{i-1}\} \tag{3.1}$$

Define $\mathbf{P}\{b_t|a_t\}$ to be the conditional matrix probability (MP) of the output symbol b_t given the input symbol a_t, as a matrix whose ij-th element is $Pr\{b_t, s_t = j|a_t, s_{t-1} = i\}$. Likewise, define $\mathbf{P}\{\mathbf{b_1^t}|\mathbf{a_1^t}\}$ to be the conditional MP of the output sequence $\mathbf{b_1^t}$ given the input sequence $\mathbf{a_1^t}$. Then $\mathbf{P}\{\mathbf{b_1^t}|\mathbf{a_1^t}\}$ is given by

$$\mathbf{P}\{\mathbf{b_1^t}|\mathbf{a_1^t}\} = \mathbf{P}\{\mathbf{b_1}|\mathbf{a_1}\}\mathbf{P}\{\mathbf{b_2}|\mathbf{a_2}\}\cdots\mathbf{P}\{\mathbf{b_t}|\mathbf{a_t}\} = \prod_{i=1}^{t} \mathbf{P}\{\mathbf{b_i}|\mathbf{a_i}\} \tag{3.2}$$

The conditional probability $Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}, s_0\}$ of receiving $\mathbf{b_1^t}$, given the initial state s_0 and the input sequence $\mathbf{a_1^t}$ is the sum over s_t of $Pr\{\mathbf{b_1^t}, s_t|\mathbf{a_1^t}, s_0\}$

$$\mathbf{p}\{\mathbf{b_1^t}|\mathbf{a_1^t}\} = \mathbf{P}\{\mathbf{b_1^t}|\mathbf{a_1^t}\}\mathbf{1} \qquad \mathbf{1} = [1\ 1\ \cdots\ 1]' \tag{3.3}$$

where $\mathbf{p}\{\mathbf{b_1^t}|\mathbf{a_1^t}\}$ is a matrix column whose elements are the probabilities $Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}, s_0\}$, i.e.,

$$\mathbf{p}\{\mathbf{b_1^t}|\mathbf{a_1^t}\} = [Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}, s_0 = 1\}\ Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}, s_0 = 2\}\ \cdots\ Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}, s_0 = u\}]' \tag{3.4}$$

Therefore, the conditional probability $Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}\}$ of receiving $\mathbf{b_1^t}$ given the channel input $\mathbf{a_1^t}$ is given by

$$Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}\} = \sum_{s_0=1}^{u} Pr\{s_0\} Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}, s_0\} \tag{3.5}$$

which can be written as

$$Pr\{\mathbf{b_1^t}|\mathbf{a_1^t}\} = \boldsymbol{\pi} \prod_{i=1}^{t} \mathbf{P}\{\mathbf{b_i}|\mathbf{a_i}\}\mathbf{1} \tag{3.6}$$

where $\boldsymbol{\pi} = [Pr\{s_0 = 1\}\ Pr\{s_0 = 2\}\ \cdots\ Pr\{s_0 = u\}]$ is a row vector of the channel's initial state probabilities.

Therefore, a FSC can be described as $\{S, A, B, \boldsymbol{\pi}, \mathbf{P}\{b|a\}\}$, where S is the channel state space, A is its input alphabet, B is its output alphabet, $\boldsymbol{\pi}$ is the inital state probability vector, and $\mathbf{P}\{b|a\}$ is the conditional MP of the output symbol $b \in B$ given the input symbol $a \in A$.

3.2 Hidden Markov Models

Consider an autonomous FSC[2]. An autonomous FSC is described by its alphabet A, the state space S, the initial state probabilities $\boldsymbol{\pi}$, and the MP $\mathbf{P}\{a\}$ whose elements are the probabilities $Pr\{a, j|i\}$ of transferring from the channel state i to state j and producing the output symbol a.

The probability of producing a sequence $\mathbf{a_1^t}$ takes the form [60]

$$Pr\{\mathbf{a_1^t}\} = \boldsymbol{\pi} \prod_{i=1}^{t} \mathbf{P}\{a_i\}\mathbf{1} \tag{3.7}$$

Elements of the MP $\mathbf{P}\{a\}$ can be written as

$$Pr\{a, j|i\} = Pr\{j|i\}Pr\{a|i, j\} \tag{3.8}$$

where the model states constitute a Markov chain[3] with the transition probability matrix

$$\mathbf{P} = [Pr\{j|i\}] = \sum_{a} \mathbf{P}\{a\} \tag{3.9}$$

The channel states are not directly observable (hidden). The observed symbol sequence is a probablistic function of the hidden states, and the symbol conditional probability (given a sequence of states) depends only on the previous state and the current state. If this conditional probability depends only on the current state, i.e, $Pr\{a|i, j\} = Pr\{a|j\}$, then the autonomous FSC is called a disrete hidden Markov model (HMM) [60].

A HMM is, therefore, a doubly stochastic process with an underlying stochastic process that is not observable, but can be observed through another set of stochastic processes that produce the sequence of observed symbols. In general, a HMM is described as $\{S, A, \boldsymbol{\pi}, \mathbf{P}, \mathbf{F}\}$, where S is the set of its hidden states, A is the set of its observations, $\boldsymbol{\pi}$ is the state initial probability vector, $\mathbf{P}$ is the state transition probability matrix, and the state observation probability matrix, $\mathbf{F}$, gives the probability of observing a symbol $a \in A$ in any state $s \in S$.

If the underlying Markov chain is stationary, and the matrix $\mathbf{P}$ is regular, then the state initial probability vector can be found as a solution to the system [73]

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi} \qquad \boldsymbol{\pi}\mathbf{1} = 1 \tag{3.10}$$

Therefore, a stationary HMM can be defined by the collection of the MPs $\mathbf{P}\{a\}$ only, since the initial state probability vector can be determined as a unique solution to (3.10).

[2] A FSC whose output does not depend on its input is called an autonomous FSC.

[3] A sequence of random variables $y_0, y_1, y_2, \ldots$ is called a finite Markov chain if the probability $Pr\{y_t|y_0, y_1, \ldots, y_{t-1}\} = Pr\{y_t|y_{t-1}\}$ for any $t \in \{1, 2, \ldots\}$.

3.3 Markov Fading Channel

Multipath or scattering in a mobile wireless communications channel causes the received signal to spread or disperse in time, i.e., scattering causes the transmitted rays to arrive at the receiver with different time delays. This spreading phenomenon introduces interference among the contiguous received symbols. The amount of time dispersion or the extent of interference depends on the bandwidth, B_s, and the symbol period, T_s, of the transmitted signal. If the bandwidth, B_s, is less than the coherence bandwidth of the channel, and if the symbol period, T_s, is greater than the channel's rms delay spread, then the channel undergoes flat fading. If the converse is true, then the channel undergoes frequency-selective fading. A flat-fading channel preserves the spectral characteristics of the transmitted signal, while a frequency-selective fading channel introduces inter-symbol interference (ISI) in the transmitted signal. Besides multipath, Doppler effects due to transmitter, receiver, and/or channel mobility give rise to correlated fading or a time-varying channel impulse response (CIR). Once again, the extent of correlation depends on the bandwidth and the symbol period of the transmitted signal. If the bandwidth, B_s, is less than the Doppler bandwidth, and if the symbol period, T_s, is greater than the channel's coherence time, then the channel experiences fast fading, i.e., the CIR varies rapidly within the symbol duration. If the converse is true, then the channel experiences slow fading.

Both multipath and Doppler effects give rise to time-varying amplitude and phase fluctuations in the received signal. Moreover, the interference due to multipath and/or correlated fading due to Doppler effects, cause the wireless mobile channel to exhibit memory. It is this inherent memory that allows us to model and characterize mobile wireless channels with HMMs. In this thesis, we shall only consider a slow, flat fading channel. It is well known that the received envelope in a flat fading channel is Rayleigh distributed [74], i.e.,

$$f(c) = \frac{c}{\sigma^2}\exp\left[\frac{-c^2}{2\sigma^2}\right]u(c) \tag{3.11}$$

where c is the received envelope, and σ^2 is the time-average power of the received signal. Moreover, the channel's autocorrelation function can be defined using Clarke's model [76]

$$R(\tau) = \sigma^2 J_0(2\pi f_d|\tau|) \tag{3.12}$$

where $J_0(\cdot)$ is the zero order modified Bessel function of the first kind, and f_d is the maximum Doppler frequency. Quantizing or partitioning the received envelope into several intervals enables us to construct a HMM for the fading channel. The resulting Markov fading channel can be described as $\{S, A, \boldsymbol{\pi}, \mathbf{P}, \mathbf{F}\}$, where S is the N-dimensional state space (N is the number of intervals used to partition the received envelope), A is the quantized fading amplitude alphabet, $\mathbf{P}$ is the transition probability matrix, and $\mathbf{F}$ is the probability matrix of observing $a \in A$ given $s \in S$.

Advantages of constructing HMMs for fading channels include the ability to model and simulate complex time-varying phenomenon with ease, and the ability to compute channel (fading) statistics in closed form. For instance, the autocorrelation function for a Markov fading channel be computed as [72]

$$R(0) = \boldsymbol{\pi}\mathbf{E}(a^2)\mathbf{1} \qquad R(\tau) = \boldsymbol{\pi}\mathbf{E}(a)\mathbf{P}^{\tau-1}\mathbf{E}(a)\mathbf{1} \tag{3.13}$$

where the matrix expectations $\mathbf{E}(a)$ and $\mathbf{E}(a^2)$ are given by

$$\mathbf{E}(a) = \int_{-\infty}^{\infty} a\mathbf{P}\mathbf{F}\{a\}da \qquad \mathbf{E}(a^2) = \int_{-\infty}^{\infty} a^2\mathbf{P}\mathbf{F}\{a\}da \tag{3.14}$$

Moreover, $\mathbf{1}$ is a column vector of ones, and the product $\mathbf{PF}\{a\}$ is the matrix probability density function of the fading amplitude a. Likewise, the fade duration distributions can also be computed in closed from. Let

$$\mathbf{P_a} = \int_{R}^{\infty} \mathbf{P}\mathbf{F}\{a\}da \tag{3.15}$$

be the matrix probability of the fading level to be above R, and

$$\mathbf{P_b} = \int_{-\infty}^{R} \mathbf{P}\mathbf{F}\{a\}da \tag{3.16}$$

be the matrix probability of the fading level to be below R. Then, the probability that the fade duration is equal to τ can be written as

$$p_\tau = \frac{\boldsymbol{\pi}\mathbf{P_a}\mathbf{P_b}^{\tau}\mathbf{P_a}\mathbf{1}}{\boldsymbol{\pi}\mathbf{P_a}\mathbf{1}} \tag{3.17}$$

The mean duration of fades, $\bar{\tau}$, and the mean duration between fades, $\bar{\nu}$, can be expressed as

$$\bar{\tau} = \frac{\boldsymbol{\pi}\mathbf{P_b}\mathbf{1}}{\boldsymbol{\pi}\mathbf{P_a}\mathbf{1}} \tag{3.18}$$

$$\bar{\nu} = \frac{\boldsymbol{\pi}\mathbf{P_a}\mathbf{1}}{\boldsymbol{\pi}\mathbf{P_b}\mathbf{1}} \tag{3.19}$$

Likewise, the level crossing rate can be found using

$$N_R = 2\boldsymbol{\pi}\mathbf{P_a}\mathbf{P_b}\mathbf{1} \tag{3.20}$$

3.4 Markov Model Parameter Estimation

The number of levels, N, and their distribution (uniform or non-uniform) used to quantize the Rayleigh envelope is arbitrary. However, the choice of N is application dependent, and will vary with the goodness-of-fit demanded from the resulting HMM. Likewise, the quantized fading amplitudes can be arbitrarily chosen (provided that the chosen values lie within the corresponding quantization regions), but are typically set equal to the centroid of the corresponding quantization region. Once the received envelope has been quantized, the HMM is constructed by estimating the initial state probability vector $\boldsymbol{\pi}$, the transition probability matrix $\mathbf{P}$, and the observation probability matrix $\mathbf{F}$. Several different approaches have been proposed for estimating the aforementioned HMM parameters. We shall consider three popular techniques, viz., the analytic approach, the Monte Carlo method, and the Baum-Welch algoritm.

3.4.1 Analytic Approach

For a Rayleigh faded channel, the received signal-to-noise ratio (SNR) is exponentially distributed [75], i.e,

$$f(\gamma) = \frac{1}{\bar{\gamma}} e^{-\gamma/\bar{\gamma}} \qquad \gamma \geq 0 \tag{3.21}$$

where $\bar{\gamma}$ is the average SNR. If f_d is the maximum Doppler frequency, then the level crossing rate is given by

$$N_a = \sqrt{\frac{2\pi\gamma}{\bar{\gamma}}} f_d e^{-\gamma/\bar{\gamma}} \tag{3.22}$$

Let $0 = A_0 < A_1 < A_2 < \cdots < A_K = \infty$ be the quantization intervals for the received SNR. Then the Rayleigh fading channel is said to be in state s_k, $k \in \{0, 1, 2, \cdots, K-1\}$, if the received SNR is in the interval $[A_k, A_{k+1})$. Moreover, the steady state probability[4] is given by [71]

$$\pi_k = \int_{A_k}^{A_{k+1}} f(\gamma) d\gamma = e^{-A_k/\bar{\gamma}} - e^{-A_{k+1}/\bar{\gamma}} \tag{3.23}$$

Assume that the Rayleigh fading channel is slow enough that the received SNR remains at a certain level for the time duration of the channel symbol. Furthermore, assume that the

[4]Since the Markov chain is stationary and regular, the steady state probabilities equal the initial state probabilities.

channel states associated with consecutive symbols are neighboring states, then the transition probability from state s_j to state s_k is

$$p_{j,k} = 0 \qquad \forall |j-k| > 1 \tag{3.24}$$

If R_s is the transmitted symbol rate, then on average

$$R_s^k = R_s \times \pi_k \tag{3.25}$$

symbols per second are transmitted during which the channel is in state s_k. Due to the slow fading assumption, we can conclude that the level crossing rate at A_k and/or A_{k+1} is much smaller than the value of R_s^k. The transition probability $p_{k,k+1}$ can then be approximated by the ratio of the expected level crossing at A_{k+1} divided by the average symbols per second the SNR falls in the interval associated with the state s_k. Likewise, the transition probability $p_{k,k-1}$ can be approximated by the ratio of the expected level crossing at A_k divided by the average symbols per second the SNR falls in the interval associated with the state s_k.

Let N_k, $k \in \{1, 2, 3, \cdots, K-1\}$, be the expected number of times per second the received SNR passes downward across the threshold A_k. Then the transition probabilities are given by [71]

$$p_{k,k+1} \approx \frac{N_{k+1}}{R_s^k}, \qquad k \in \{0, 2, 3, \cdots, K-2\} \tag{3.26}$$

and

$$p_{k,k-1} \approx \frac{N_k}{R_s^k}, \qquad k \in \{1, 2, 3, \cdots, K-1\} \tag{3.27}$$

Furthermore, the transition probabilities $p_{0,0}$, $p_{K-1,K-1}$, and $p_{k,k}$ are given by [71]

$$\begin{aligned} p_{0,0} &= 1 - p_{0,1} \\ p_{K-1,K-1} &= 1 - p_{K-1,K-2} \\ p_{k,k} = 1 - p_{k,k+1} - p_{k,k+1}, &\qquad k \in \{1, 2, 3, \cdots, K-2\} \end{aligned} \tag{3.28}$$

Since their exist a one-to-one correspondence between the states and the quantized fading amplitudes, the observation probability matrix equals a $K \times K$ identity matrix. Furthermore, the initial state probability vector can be computed as a solution to (3.10).

Although simple, this first-order HMM model approximation of a Rayleigh fading channel is valid only for small quantization intervals [72]. To approximate the Bessel function is (3.12) requires a Markov chain with large memory. The analytic approach is not feasible, since the number of states in the Markov chain grows exponentially with the process memory. However, the analytic approach allows us to construct *simple* HMMs that can be used in proof-of-concept type analysis of communication systems.

3.4.2 Monte Carlo Method

The Monte Carlo method offers an alternate solution to the analytic approach for estimating HMMs for Rayleigh fading channels. If the Rayleigh envelope is quantized into N intervals, then the transition probability between the HMM states is given by [66]

$$p_{i,j} = \frac{n_{i,j}}{n_i} \tag{3.29}$$

where $i \in \{1, 2, \cdots, N\}$, $n_{i,j}$ is the number of transitions from state s_i to state s_j, and n_i is the total number of visits to state s_i, i.e.,

$$n_i = \sum_{j=1}^{N} n_{i,j} \tag{3.30}$$

The resulting HMM assumes that the quantized fading amplitude from the current state can jump only to the adjacent states, i.e., $p_{i,j} = 0$ if $|i - j| > 1$. Therefore, if $p_{i,i}$ is large, the Monte Carlo method allows us to model slowly varying fading processes. As in the previous section, the observation probability matrix equals a $K \times K$ identity matrix, and the initial state probability vector can be computed as a solution to (3.10).

The Monte Carlo method proposed in [66] does not approximate the state duration distributions accurately . However, the technique can be improved by splitting each state into two sub-states, representing increasing and decreasing fading amplitude values [70]. Although the positive-slope states may assume identical fading amplitude values as their negative-slope counterparts, the two sets of states are distinct in that the upper branch states can be visited only when the fading amplitude is increasing, whereas the lower branch states can be visited only when the fading amplitude is decreasing.

The accuracy of the two aforementioned methods depends on the selection of the quantization levels. The quantization levels must satisfy two conditions. First, the transition probabilities, $p_{i,j}$, should be close to zero for $|i - j| > 1$. Second, the original model's state duration distributions must be close to an exponential distribution[5] [72]. However, these two conditions are difficult to satisfy. For instance, in order to fit the exponential state duration distribution, the number of quantization levels must be large, but in this case the transition probabilities to adjacent states are not negligibly small.

The Monte Carlo method is relatively simple, but inaccurate especially when it is used to estimate small values, and when the amount of statistical data is limited. However, the Monte Carlo method allows fast estimation of HMMs which can be used as initial approximations in more robust HMM estimation algorithms.

[5]The state durations of a Markov chain have an exponential probability distribution.

3.4.3 Baum-Welch Algorithm

One of the most powerful methods of approximating a stochastic process with a HMM consists of fitting the multi-dimensional probability distributions of a HMM to that of the original process. The HMM parameters can be obtained as those which minimize the Kullback-Leibler divergence [72], i.e.,

$$\theta = \arg\min K(f||p_\theta) \tag{3.31}$$

where

$$K(f||g_\theta) = \int_{X^k} f(\mathbf{x_1^k}) log \frac{f(\mathbf{x_1^k})}{g_\theta(\mathbf{x_1^k})} d\mathbf{x_1^k} \tag{3.32}$$

and $f(\cdot)$ is the multi-dimensional probability distribution of the stochastic process (fading envelope), and $g_\theta(\cdot)$ are the probability densities of the HMM output. The minumum in (3.32) can be obtained iteratively via the expectation-maximization (EM) algorithm [60], [77]. The special case of the EM algorithm that is used to fit HMMs to a stochastic process is called the Baum-Welch algorithm (BWA) [60], [78].

To estimate a HMM using the BWA, the received envelope is quantized into N levels. These N levels are said to partition the state space S of the HMM into N disjoint sub-sets. Each of these sub-sets (or quantized levels) are further partitioned into u states, such that the dimension of S, dim S = dim $N \times$ dim u. The N levels are selected in such a way that the sequence of quantized fading amplitudes can be approximated by a semi-Markov process[6]. Since we assumed slow fading, the fade levels of two consecutive received symbols belong either to the same quantization interval, or to the adjacent intervals. Hence, the semi-Markov process can be described by the $N \times N$ transition probability matrix

$$\mathbf{P} = \begin{bmatrix} p_{1,1} & p_{1,2} & 0 & 0 & \cdots & 0 & 0 \\ p_{2,1} & p_{2,2} & p_{2,3} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & p_{N,N-1} & p_{N,N} \end{bmatrix} \tag{3.33}$$

where the individual transition probabilities, $p_{i,j}$, can be computed using either the analytic approach or the Monte Carlo method.

It is well known that the fade level durations of a Rayleigh channel have geometrical distributions [60]. It is also well known that semi-Markov processes with real transitions[7] have

[6] In a Markov process, transitions occur at every time instant. Contrariwise, in a semi-Markov process, the interval between successive transitions may be several time units long. Furthermore, this transition time depends on the state presently occupied, and on the state to which the next transition will be made.

[7] A real transition requires a change in the state indices after a transition.

geometric holding times, while those with virtual transitions[8] have holding times equal to one unit [73]. Hence, the transition probability matrix in (3.33) needs to be modified to allow only real transitions, i.e.,

$$\mathbf{P} = \begin{bmatrix} 0 & \bar{p}_{1,2} & 0 & 0 & \cdots & 0 & 0 \\ \bar{p}_{2,1} & 0 & \bar{p}_{2,3} & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \bar{p}_{N,N-1} & 0 \end{bmatrix} \tag{3.34}$$

where the off-diagonal transition probabilities are given by

$$\bar{p}_{i,j} = \frac{p_{i,j}}{1 - p_{i,i}} \qquad i, j \in \{1, 2, \ldots, N\} \tag{3.35}$$

To construct the HMM, we need to explicitly match the state duration distributions of the quantized fading process to the state duration distributions of the HMM. Since the state durations of a Rayleigh channel have a Gamma distribution, they can be approximated using phase-type (PH) matrix-geometric distributions [60]

$$g_i(x) = \boldsymbol{\mu_i} \mathbf{A}_i^{x-1} \mathbf{B}_i \qquad x = 1, 2, \ldots \tag{3.36}$$

where $i \in \{1, 2, \ldots, N\}$, x is the state duration, $g(x)$ is the probability of observing x, $\boldsymbol{\mu_i}$ is a $1 \times u$ row vector, $\mathbf{B}_i$ is a $u \times 1$ column vector, and $\mathbf{A}_i$ is a $u \times u$ square matrix such that

$$\mathbf{E}_i = \begin{bmatrix} \mathbf{A}_i & \mathbf{B}_i \\ \boldsymbol{\mu_i} & 0 \end{bmatrix} \tag{3.37}$$

is a stochastic matrix[9]. The vectors $\boldsymbol{\mu_i}$ and $\mathbf{B}_i$ are the ingress and the egress[10] state probability vectors for quantization level i. The matrix $\mathbf{A}_i$ describes the transition probabilities between the u states of the i-th quantization level. Hence, the probability of observing the i-th level (or the i-th quantized fading amplitude) in these u states is equal to one. Furthermore, since $\mathbf{E}_i$ is a stochastic matrix, it follows that $\boldsymbol{\mu_i}\mathbf{1} = 1$ and $\mathbf{B}_i = (\mathbf{I} - \mathbf{A}_i)\mathbf{1}$, where $\mathbf{1}$ is $[1\ 1\ \cdots\ 1]'$, and $\mathbf{I}$ is the identity matrix [60].

Let η_{a_i} be the total number of successive observations of the quantized fading amplitude a_i, associated with the i-th quantization level. Let O be the quantized Rayleigh envelope recorded over some observation window. Then, O can be expressed as

$$O = \eta_{a_i} \eta_{a_j} \eta_{a_k} \cdots \qquad i, j, k \in \{1, 2, \ldots, N\} \tag{3.38}$$

[8] A virtual transition does not require a change in the state indices after a transition.

[9] Each row in a stochastic matrix sums to one.

[10] When a HMM enters or exits the i-th quantization level, it does so by transitioning to or form one of the u states that belong to that level. The ingress state probability vector gives the probability of being in one of the u states when the HMM enters the i-th quantization level. Conversely, the egress state probability vector gives the probability of being in one of the u states when the HMM exits the i-th quantization level.

Let the observation sequence, $\mathbf{x}_1^T$, be the collection of all terms in (3.38) that correspond to the i-th quantization level. Let T be the length of this sequence. Then, $\mathbf{x}_1^T$ gives the run-length distribution of the state durations for quantization level i. Given the observation sequence, the aforementioned PH distribution parameters for the i-th quantization level can be estimated iteratively using the BWA [60]

$$\mu^{n+1}(i) = \frac{\mu^n(i)}{T} \sum_{m=1}^{T} \frac{\beta^\eta(i, x_m)}{g^n(x_m)} \tag{3.39}$$

$$A^{n+1}(i,j) = A^n(i,j) \frac{\sum_{m=1}^{T} \sum_{k=1}^{x_m - 1} \alpha^n(i, k-1) \beta^n(j, x_m - k) / g^n(x_m)}{\sum_{m=1}^{T} \sum_{k=1}^{x_m} \alpha^n(i, k-1) \beta^n(i, x_m - k + 1) / g^n(x_m)} \tag{3.40}$$

$$\mathbf{B}^{n+1} = (\mathbf{I} - \mathbf{A}^{n+1})\mathbf{1} \tag{3.41}$$

where $\mu(i)$ is the i-th element in the row vector $\boldsymbol{\mu}$, $A(i,j)$ is the ij-th element in the matrix $\mathbf{A}$, n is the iteration index, and $\alpha(i,k)$ and $\beta(i,k)$ are defined as

$$\alpha(i,k) = \boldsymbol{\mu}\mathbf{A}^k\mathbf{e}_i^{'} \qquad \beta(i,k) = \mathbf{e}_i\mathbf{A}^{k-1}\mathbf{B} \tag{3.42}$$

where $\mathbf{e}_i$ is a row vector whose i-th element is a one and the rest of the elements are zero. The PH distribution parameters are initialized such that $\mu^0(i) = 1/u$, $A^0(i,i) > A^0(i, i \pm 1) > A^0(i, i \pm 2) > \cdots$, and the initial values for the column vector $\mathbf{B}$ can be obtained as a solution to (3.41). It is important to note that elements initialized as zero will always retain their initial value[11].

For large k, both α and β approach zero geometrically fast. Hence, a direct implementation of the BWA yields an unstable and inaccurate algorithm. However, more stable and elegant implementations of the BWA can be realized by using either the forward-backward algorithm or the uni-directional algorithm [60], [79] (in this thesis, we shall only consider the forward-backward algorithm).

After estimating the PH distribution parameters for each state, the semi-Markov process can be transformed into a HMM whose $S \times S$ transition probability matrix is given by [60]

$$\mathbf{P} = \begin{bmatrix} \mathbf{A}_1 & \bar{p}_{1,2}\mathbf{B}_1\boldsymbol{\mu}_2 & 0 & 0 & \cdots & 0 & 0 \\ \bar{p}_{2,1}\mathbf{B}_2\boldsymbol{\mu}_1 & \mathbf{A}_2 & \bar{p}_{2,3}\mathbf{B}_2\boldsymbol{\mu}_3 & 0 & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \cdots & \bar{p}_{N,N-1}\mathbf{B}_N\boldsymbol{\mu}_{N-1} & \mathbf{A}_N \end{bmatrix} \tag{3.43}$$

[11]This fact may be useful in constructing HMMs with a pre-defined structure (for instance, in constructing left-to-right or right-to-left HMMs). However, in our case we will not make any assumption about the structure of our model.

Furthermore, the initial state probability vector $\boldsymbol{\pi}$ can be found as a solution to (3.10), and the $S \times S$ observation probability matrix is given by

$$\mathbf{F} = \begin{bmatrix} \mathbf{I}_1 & 0 & \cdots & 0 \\ 0 & \mathbf{I}_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \mathbf{I}_N \end{bmatrix} \tag{3.44}$$

where $\mathbf{I}_i$, $i \in \{1, 2, \ldots, N\}$, is a $u \times u$ identity matrix.

3.4.3.1 Forward-Backward Implementation

The forward-backward algorithm (FBA) facilitates a computationally efficient trellis based implementation of the BWA. The trellis consists of u states, where the transition probabilities between the states are given by the matrix $\mathbf{A}$. The length of the trellis, x_m, is the magnitude of the m-th element in the observation sequence $\mathbf{x}_1^T$. Hence, the FBA is implemented T times (for each element in the observation sequence) per BWA iteration. Let $\mathbf{y}$ be a training sequence, such that $\mathbf{y} = \{1, 2, \ldots, x_m\}$. Furthermore, let $\boldsymbol{\mu}^n$, $\mathbf{A}^n$, and $\mathbf{B}^n$ be the PH distribution parameters estimated during the n-th iteration of the BWA.

The forward variable, $\alpha(i, k)$, is defined as the probability of observing the partial training sequence $\mathbf{y}_1^k = \{1, 2, \ldots, k\}$, $k \leq x_m$, and the state s_i, $i \in \{1, 2, \ldots, u\}$, i.e.,

$$\alpha(i, k) = Pr\{1, 2, \ldots, k, s_i\} \tag{3.45}$$

For the $(n+1)$-th iteration of the BWA, the α probabilities are initialized as

$$\alpha(i, 0) = \mu^n(i) \tag{3.46}$$

and are computed using a forward recursion through the trellis

$$\alpha(j, k) = \sum_{i=1}^{u} \alpha(i, k-1) A^n(i, j) \qquad k \in \{1, 2, \ldots, x_m - 1\} \tag{3.47}$$

where $j \in \{1, 2, \ldots, u\}$. As mentioned above, the α probabilities approach zero geometrically fast for large x_m. They must, therefore, be normalized to maintain the accuracy and the stability of the FBA. Hence,

$$\alpha(j, k) = \frac{\sum_{i=1}^{u} \alpha(i, k-1) A^n(i, j)}{\sum_{j=1}^{u} \sum_{i=1}^{u} \alpha(i, k-1) A^n(i, j)} \qquad k \in \{1, 2, \ldots, x_m - 1\} \tag{3.48}$$

The backward variable, $\beta(i,k)$, is defined as the probability of observing the partial training sequence $\mathbf{y}_{k+1}^{x_m} = \{k+1, k+2, \ldots, x_m\}$, $k > 0$, given the state s_i, i.e.,

$$\beta(i,k) = Pr\{k+1, k+2, \ldots, x_m | s_i\} \tag{3.49}$$

For the $(n+1)$-th iteration of the BWA, the β probabilities are initialized as

$$\beta(i, x_m - 1) = B^n(i) \tag{3.50}$$

and are computed and normalized using a backward recursion through the trellis

$$\beta(j,k) = \frac{\sum_{i=1}^{u} \beta(i,k+1) A^n(j,i)}{\sum_{j=1}^{u} \sum_{i=1}^{u} \alpha(i,k-1) A^n(i,j)} \qquad k \in \{x_m - 2, x_m - 3, \ldots, 0\} \tag{3.51}$$

Note that both the forward and the backward probabilities are normalized by the same scaling factor. Let $\gamma(i,k)$ be the probability of being in state s_i, given the entire training sequence. Hence,

$$\gamma(i,k) = \frac{\alpha(i,k)\beta(i,k)}{g(x_m)} \qquad k \in \{0, 1, \ldots, x_m - 1\} \tag{3.52}$$

Therefore, the expected number of transitions made from state s_i is given by $\sum_{k=1}^{x_m} \gamma(i,k)$. Likewise, let $\xi(i,j,k)$ be the probability of transferring from state s_i to state s_j, given the entire training sequence. Then, $\xi(i,j,k)$ can be expressed as

$$\xi(i,j,k) = \frac{\alpha(i,k) A^n(i,j) \beta(j,k+1)}{g(x_m)} \qquad k \in \{0, 1, \ldots, x_m - 2\} \tag{3.53}$$

and the expected number of transitions made from state s_i to state s_j is given by $\sum_{k=1}^{x_m - 1} \xi(i,j,k)$.

Once the FBA has been implemented on each element in the observation sequence, $\mathbf{x}_1^T$, the PH distribution parameters can be re-estimated as

$$\mu^{n+1}(i) = \frac{1}{T} \sum_{m=1}^{T} \gamma(i,1) \qquad i \in \{1, 2, \ldots, u\} \tag{3.54}$$

$$A^{n+1}(i,j) = \frac{\sum_{m=1}^{T} \sum_{k=1}^{x_m - 1} \xi(i,j,k)}{\sum_{m=1}^{T} \sum_{k=1}^{x_m} \gamma(i,k)} \qquad i,j \in \{1, 2, \ldots, u\} \tag{3.55}$$

Note that the column vector $\mathbf{B}$ can be re-estimated using (3.41).

3.5 Simulation Results

The HMM parameter estimation techniques described in the previous section are used to construct a 2-state HMM, an 8-state HMM, and a 32-state HMM for a slow, flat fading Rayleigh channel. The 2-state model is constructed using the analytic approach, the 8-state model is estimated using the Monte Carlo method, and the 32-state model is estimated using the Baum-Welch algorithm.

3.5.1 2-state Model

The 2-state model (henceforth, referred to as the Gilbert model) is a first order HMM with states, *good* and *bad*, denoted by G and B. The Gilbert and the slow, flat fading Rayleigh channel are matched by choosing a level for the signal-to-noise ratio (SNR) where the channel changes states, and by matching the average duration during which the fading amplitude is below this level to the average number of time units the Gilbert channel is in the bad state.

Let γ_t be the threshold for the SNR observations. This threshold is chosen arbitrarily. However, if γ_t is chosen to be very small or very large, the channel behaviour will be close to the memoryless case [80]. If γ_t is very small, the channel will seldom be in the bad state. Moreover, if the channel does enter the bad state, it will not remain there for more than a few symbols. Alternatively, if γ_t is very large, the channel will change states more often, and the effect of remaining in the bad state will no longer be that severe.

The initial state probabilities are determined by computing the fraction of time the Rayleigh fading channel is above and below γ_t, i.e.

$$\pi_G = \int_{\gamma_t}^{\infty} f(\gamma) d\gamma = e^{-\rho^2} \tag{3.56}$$

where $f(\gamma)$ is given by (3.21), $\rho^2 = -\gamma_t/\bar{\gamma}$, and $\bar{\gamma}$ is the average SNR. Likewise

$$\pi_B = \int_{0}^{\gamma_t} f(\gamma) d\gamma = 1 - e^{-\rho^2} \tag{3.57}$$

Let $P_{G,B}$ and $P_{B,G}$ be the real transition probabilities of transferring from the good to the bad state and *vice versa*. These transition probabilities are computed by matching the average time the fading amplitude remains below a certain level to the time duration the Gilbert channel remains in the bad state, i.e.,

$$P_{G,B} = \rho f_d T_s \sqrt{2\pi} \tag{3.58}$$

and

$$P_{B,G} = \frac{\rho f_d T_s \sqrt{2\pi}}{e^{\rho^2} - 1} \tag{3.59}$$

where f_d is the maximum Doppler frequency, and T_s is the symbol duration.

Figure (3.1) shows the received SNR for a slow, flat fading Rayleigh channel recorded over an observation window of 2000 samples. The Rayleigh channel is simulated by using Clarke's model [76], a normalized fading rate f_dT_s of 0.001, and a noise variance of 0.5. For the given observation window and an SNR threshold of -5 dB, the average SNR and ρ^2 are computed as -0.32 dB and 0.3402. The corresponding initial state probability vector is

$$\boldsymbol{\pi} = \begin{bmatrix} \pi_G & \pi_B \end{bmatrix} = \begin{bmatrix} 0.7111 & 0.2884 \end{bmatrix} \tag{3.60}$$

Likewise, the corresponding transition probability matrix is

$$\mathbf{P} = \begin{bmatrix} 0.9985 & 0.0015 \\ 0.0036 & 0.9964 \end{bmatrix} \tag{3.61}$$

The simulated output SNR from a Gilbert channel with the aforementioned parameters is shown in Figure (3.2). The output symbols for the good and the bad states are (arbitrarily) set to 5 dB and -5 dB.

3.5.2 8-state Model

The slow, flat fading Rayleigh channel is simulated by using Clarke's models [76], a normalized fading rate f_dT_s of 0.01, and a sample size of 10^6. The Rayleigh envelope is quantized into eight states[12], defined by the quantization regions [0 0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4]. The centroid of each region is chosen as the quantized fading amplitude for the corresponding HMM state. Table (3.1) shows the transition probabilities of the HMM estimated using the Monte Carlo method.

k	$t_{k,k-1}$	$t_{k,k}$	$t_{k,k+1}$
0	-	0.9180	0.0820
1	0.0311	0.9208	0.0481
2	0.0417	0.9187	0.0396
3	0.0480	0.9176	0.0344
4	0.0542	0.9165	0.0293
5	0.0592	0.9144	0.0264
6	0.0688	0.9055	0.0256
7	0.0491	0.9509	-

Table 3.1: Transition probabilities for a 8-state HMM.

[12]Increasing the number of states improves the resolution of the HMM, but requires a larger sample size.

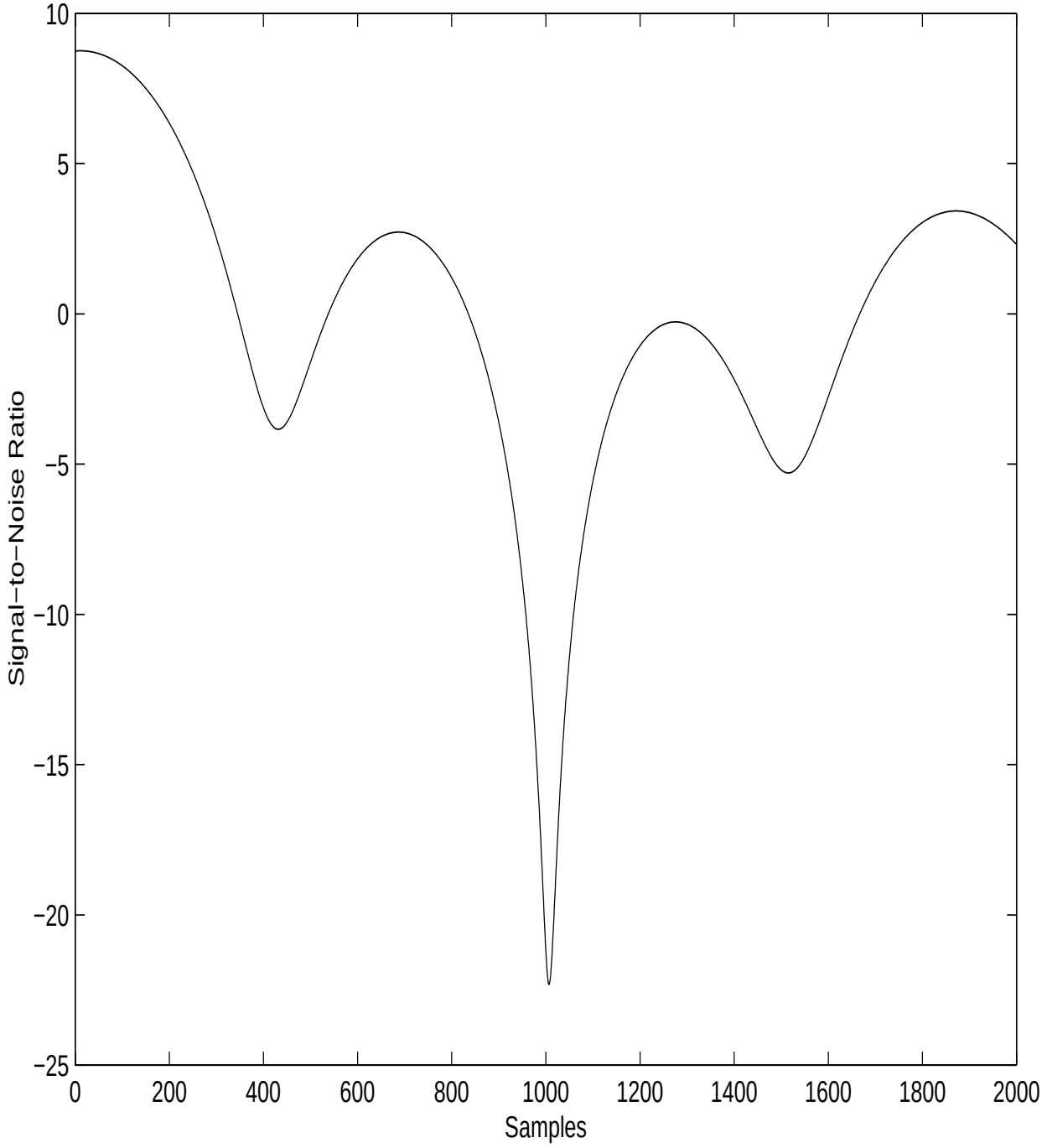

Figure 3.1: Received SNR for a slow, flat fading Rayleigh channel with a normalized fading rate $f_d T_s$ of 0.001.

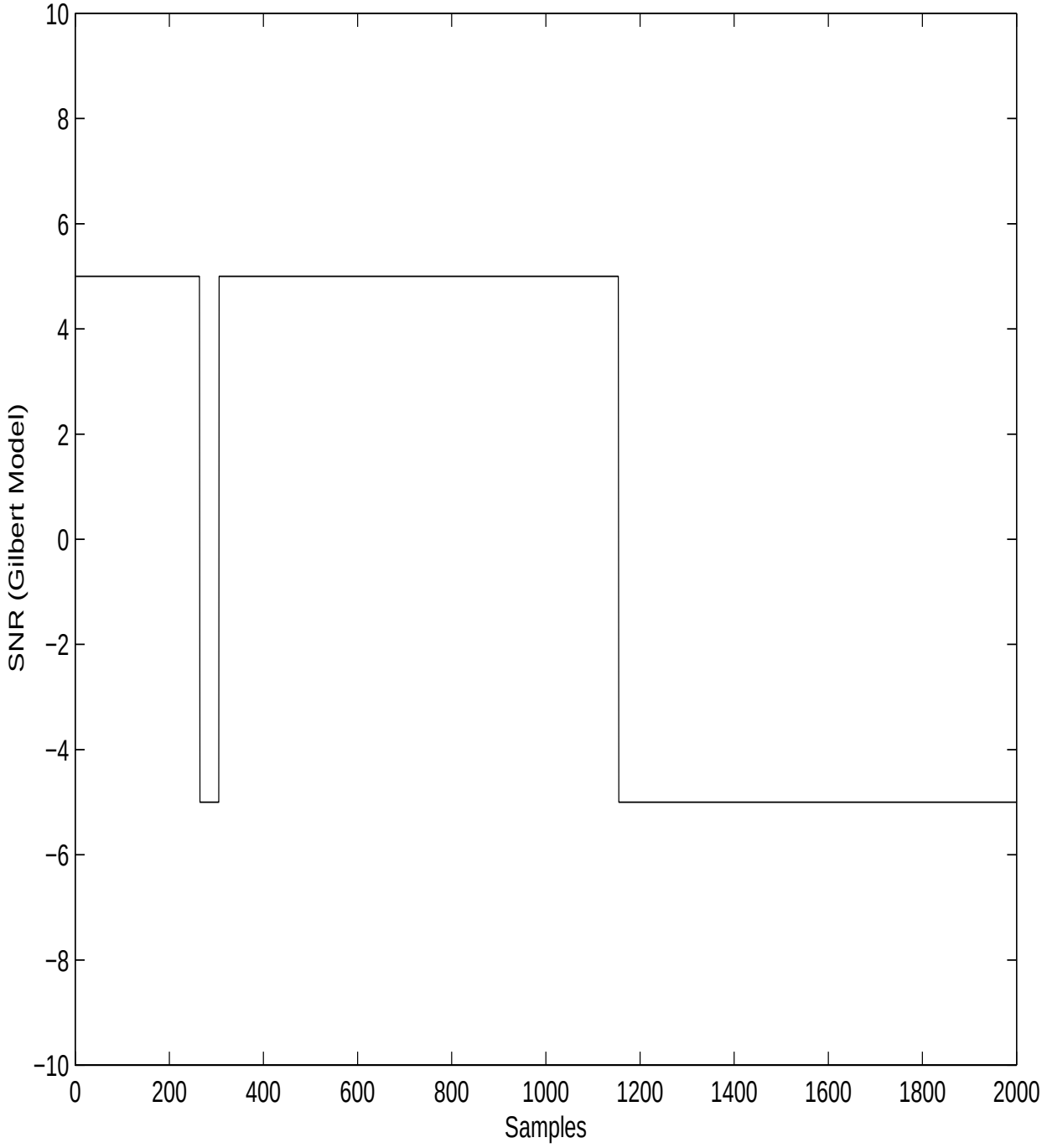

Figure 3.2: Output of a Gilbert channel matched to a slow, flat fading Rayleigh channel with a normalized fading rate $f_d T_s$ of 0.001.

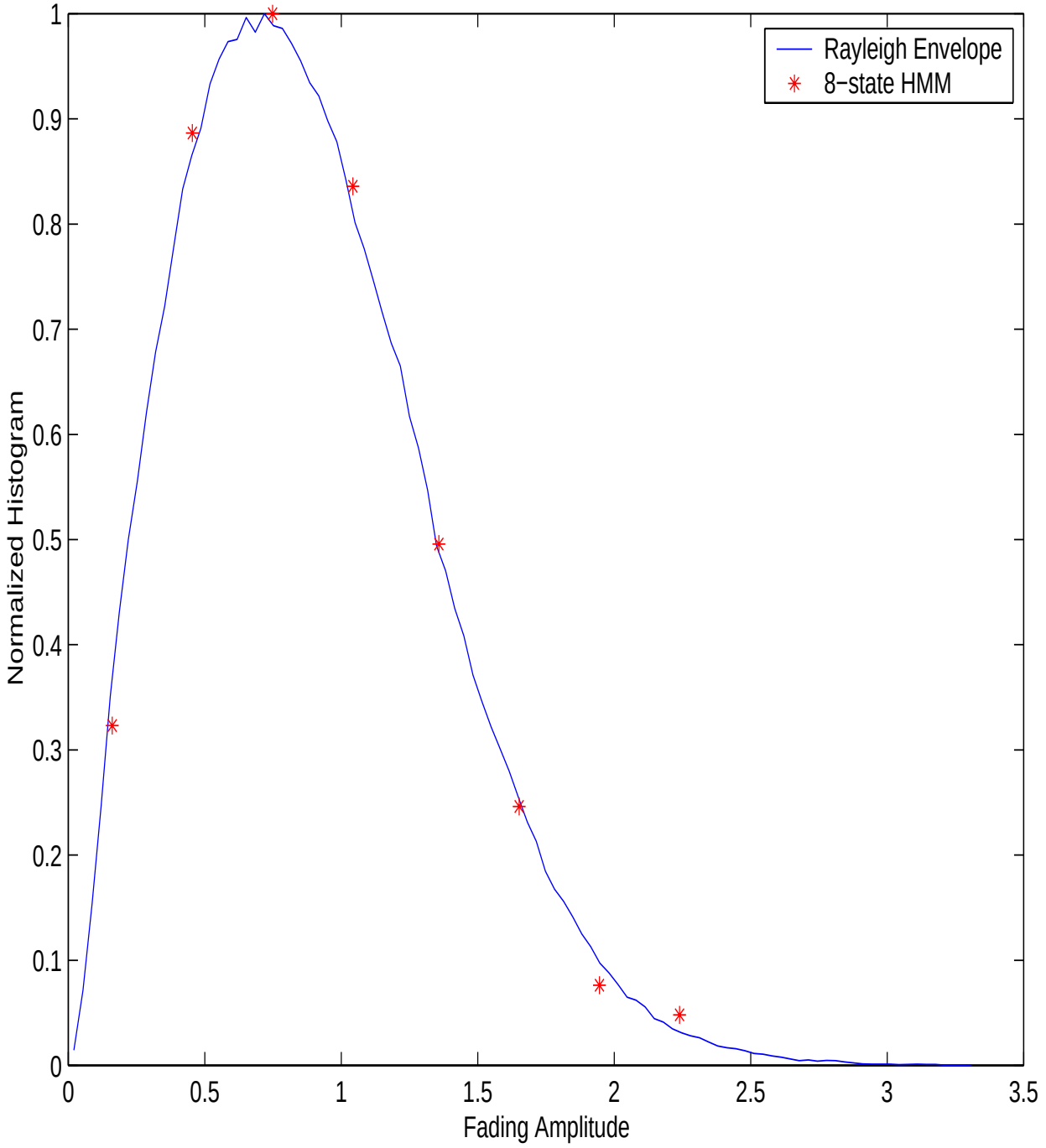

Figure 3.3: Normalized histogram of a slow, flat fading Rayleigh envelope ($f_d T_s = 0.01$) and the output of a corresponding 8-state HMM.

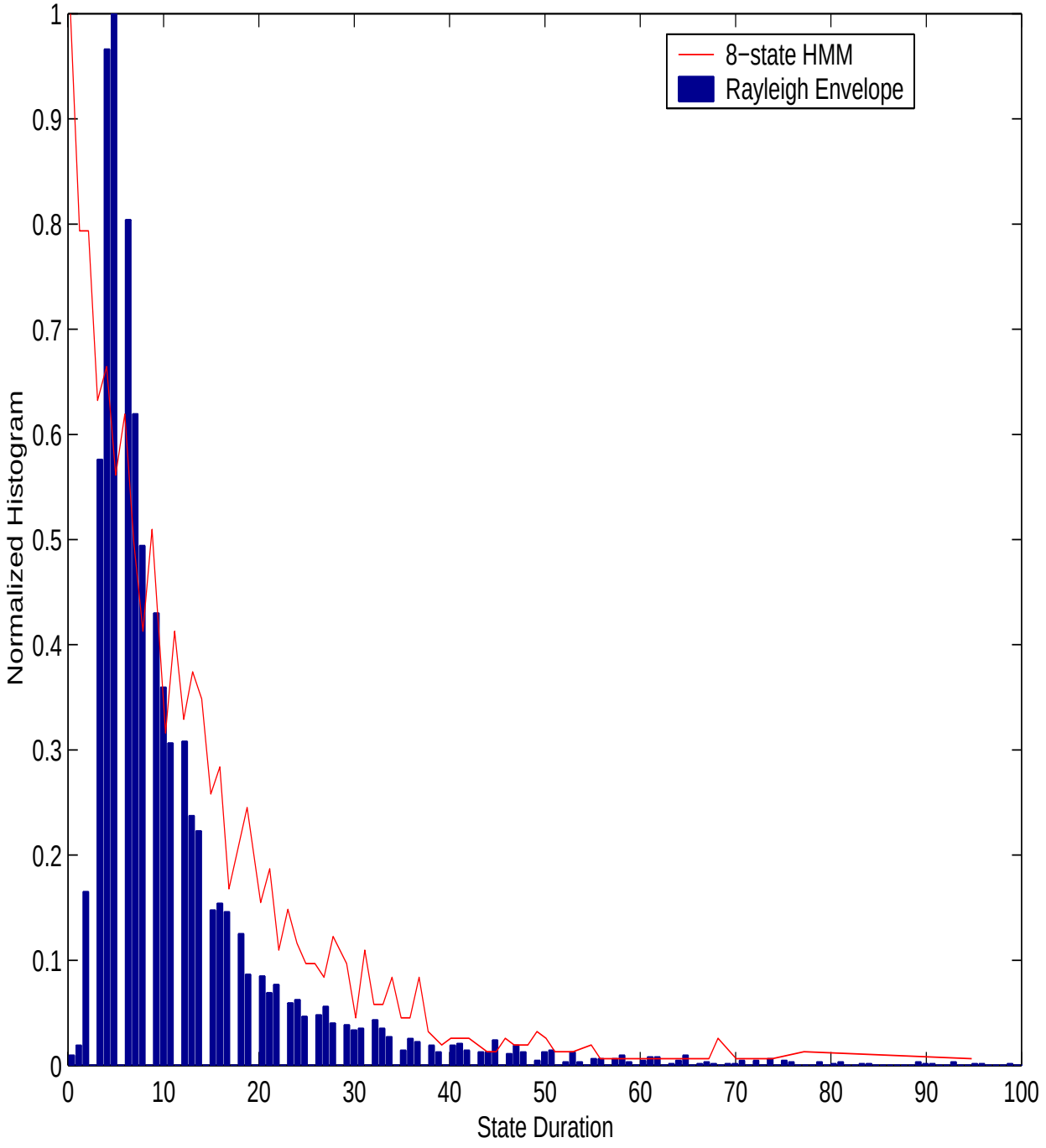

Figure 3.4: Normalized histogram of the state durations (state = 2) for a quantized Rayleigh envelope ($f_d T_s = 0.01$) and the output of a corresponding 8-state HMM.

Figure (3.3) shows the normalized histogram of the output of the 8-state HMM and the Rayleigh envelope. Likewise, Figure (3.4) shows the normalized histogram of the second ($k = 2$) quantized state's duration distribution for both the HMM and the quantized Rayleigh envelope. It is evident that even though the HMM yields a good fit for the Rayleigh distributed fading amplitudes, it fails to match the state duration distribution of the Rayleigh channel. This mismatch in state duration distributions is expected, since a Markov chain with exponentially distributed state durations cannot model a Rayleigh fading channel with Gamma distributed state durations (unless the state durations of the fading channel are explicitly matched to those of the HMM).

3.5.3 32-state Model

The 32-state HMM is constructed using the Baum-Welch algorithm. The slow, flat fading Rayleigh channel is simulated by using Clarkes models [76], a normalized fading rate $f_d T_s$ of 0.01, and a sample size of 10^6. The Rayleigh envelope is quantized into eight levels, defined by the quantization regions [0 0.3 0.6 0.9 1.2 1.5 1.9 2.3 2.9]. The transition probability matrix of the resulting semi-Markov process is estimated by using the Monte Carlo method.

i	$\bar{p}_{i,i-1}$	$\bar{p}_{i,i}$	$\bar{p}_{i,i+1}$
0	-	0	1
1	0.3928	0	0.6072
2	0.5104	0	0.4896
3	0.5827	0	0.4173
4	0.6469	0	0.3531
5	0.7566	0	0.2434
6	0.8017	0	0.1983
7	1	0	-

Table 3.2: Transition probabilities for a 8-state semi-Markov process.

Each of the eight quantization levels are further partitioned into four states. The choice of four states per quantization level was made arbitrarily. However, the optimum number of states per level can be determined by using the iterative scheme[13] proposed in [67]. The PH

[13]The HMM is first estimated by using a predetermined minimum number of states per quantization level. This HMM is tested for its goodness-of-fit. If it satisfies some *goodness* criterion, then the estimation process stops. Otherwise, the number of states are incremented and the estimation process is repeated until the HMM satisfies the stopping criterion.

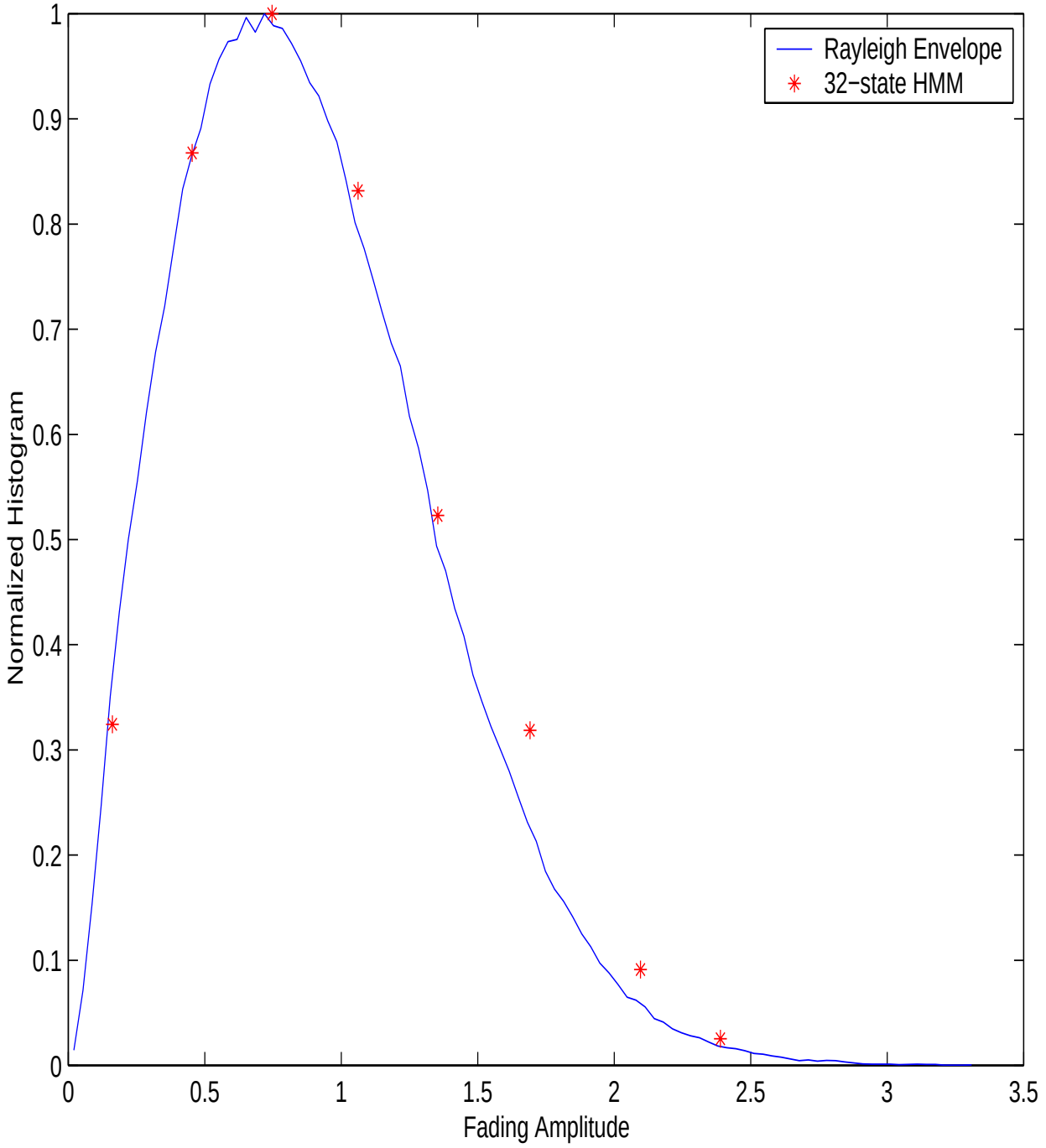

Figure 3.5: Normalized histogram of a slow, flat fading Rayleigh envelope ($f_d T_s = 0.01$) and the output of a corresponding 32-state HMM.

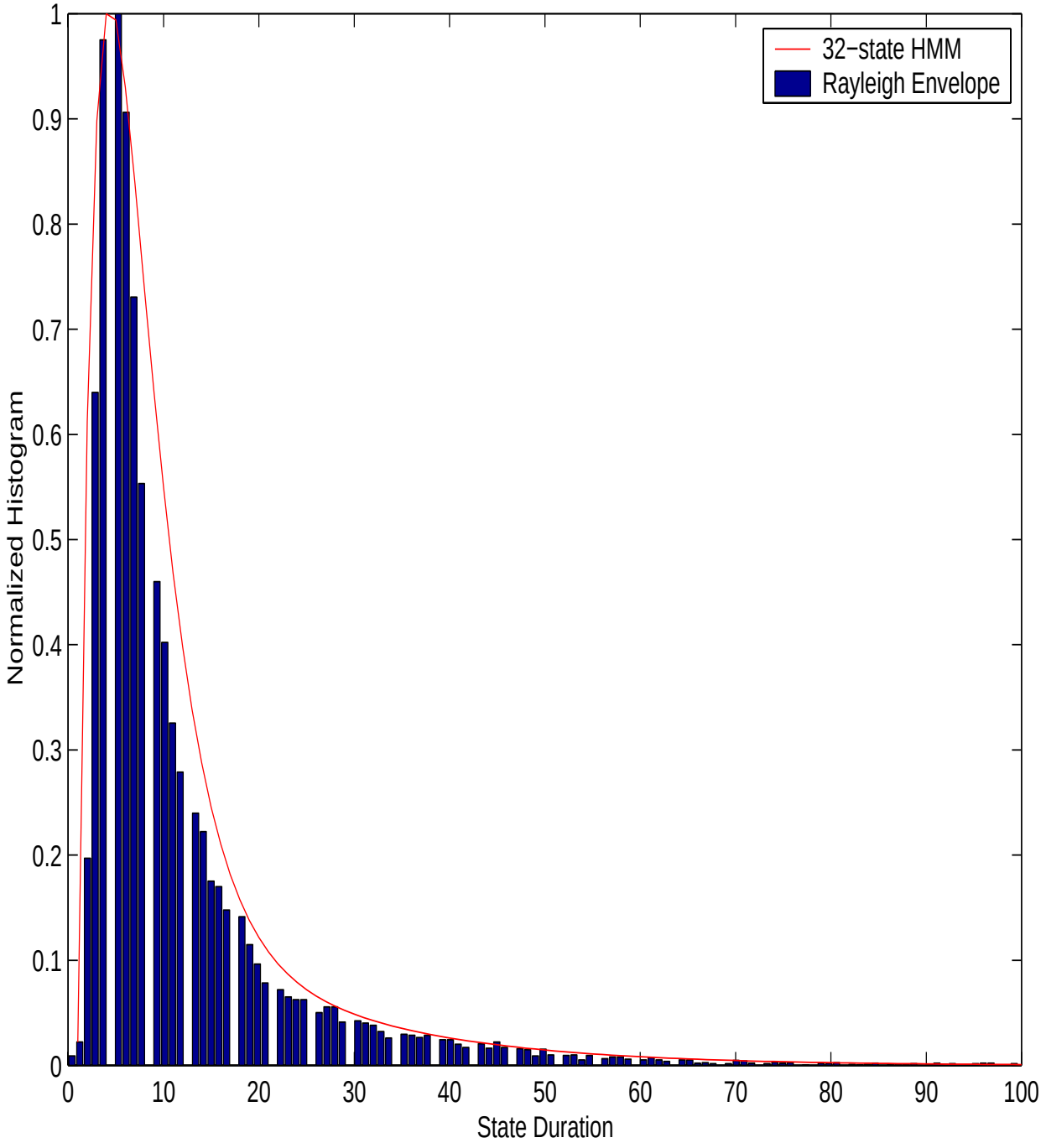

Figure 3.6: Normalized histogram of the state durations (state = 2) for a quantized Rayleigh envelope ($f_d T_s = 0.01$) and the output of a corresponding 32-state HMM.

distribution parameters for each quantization level are initialized as

$$\mathbf{E}_i = \begin{bmatrix} 0.9000 & 0.0200 & 0.0010 & 0.0040 & 0.0750 \\ 0.0200 & 0.8000 & 0.0600 & 0.0030 & 0.1170 \\ 0.0010 & 0.0100 & 0.8500 & 0.0020 & 0.1370 \\ 0.0100 & 0.0010 & 0.0020 & 0.9500 & 0.0190 \\ 0.2500 & 0.2500 & 0.2500 & 0.2500 & 0 \end{bmatrix} \qquad i \in \{1, 2, \ldots, 8\} \tag{3.62}$$

and are estimated using 500 iterations[14] of the Baum-Welch algorithm. For instance, the PH parameters for the second quantization level were estimated as

$$\mathbf{E}_2 = \begin{bmatrix} 0.9342 & 0.0239 & 0.0032 & 0.0017 & 0.0371 \\ 0.0259 & 0.7329 & 0.2401 & 0.0012 & 0.0000 \\ 0.0007 & 0.0009 & 0.7409 & 0.0010 & 0.2564 \\ 0.0130 & 0.0013 & 0.0443 & 0.9369 & 0.0045 \\ 0.0337 & 0.8076 & 2.834 \times 10^{-6} & 0.1587 & 0 \end{bmatrix} \tag{3.63}$$

Once the PH parameters for all the eight levels have been estimated, the 32-state HMM is constructed using (3.43). Figure (3.5) shows the normalized histogram of the output of the HMM and the Rayleigh envelope. The resolution of the HMM can be improved by increasing the number of quantization levels. However, doing so requires a much larger sample size, and slows the convergence rate of the Baum-Welch Algorithm. Figure (3.6) shows the normalized histogram of the second ($k = 2$) quantized state's duration distribution for both the HMM and the quantized Rayleigh envelope. It is evident that unlike the Monte Carlo method, the Baum-Welch algorithm yields a good fit for both the histogram of fading amplitudes and the histogram of state durations.

3.6 Chapter Summary

Markov models have been used for a number of applications that vary from speech processing, pattern recognition, signal estimation, equalization, and burst error simulation. Recently, Markov models have been used to model burst errors in fading channels, and more importantly the fading channel itself. This variegated use of Markov models is fueled by their ability to model time-varying systems with memory, their versatility, and their ease of implementation over other techniques.

This chapter presented an introduction to finite state channels and hidden Markov models (HMMs). The chapter described the concept and advantages of representing a slow, flat fading Rayleigh channel by a HMM. For instance, it was shown that a HMM representation of a

[14]The Baum-Welch algorithm has an extremely slow convergence rate. The number of iterations required for a level will vary with the size of the run-length (state duration) distribution sequence for that level.

fading channel facilitates an easy closed form solution to the otherwise complicated process of computing fading statistics. A HMM representation also facilitates simple and efficient hardware and/or software based simulation strategies for a fading channel. The chapter also described several techniques for estimating HMMs for a fading channel. It was shown that while the analytic and the Monte Carlo methods were simple to implement, they do not fully model the fading process, i.e., they do not accurately model the state duration distributions of the quantized fading channel. However, it was also shown that these techniques can be used to support the more robust Baum-Welch algorithm. Although computationally expensive, the Baum-Welch algorithm was shown to accurately model both the fading amplitude distribution, and the state duration distributions of the fading channel. However, the Baum-Welch algorithm has an extremely slow convergence rate, and is, therefore, not suitable for the on-line estimation of HMMs. Future studies should, therefore, investigate the incremental expected-maximization algorithm and/or the segmented K-means algorithm as alternatives to the Baum-Welch algorithm. These studies should also investigate the estimation of HMMs for fast fading channels, frequency-selective fading channels, and for fading channels where the received envelope has non-Rayleigh distributions.

Chapter 4

Channel Estimation

Digital receivers operating in time-varying fading environments require knowledge of the channel side information (SI), viz., the fading amplitude, the phase, and the noise variance for signal demodulation, equalization, iterative decoding, and for a host of other baseband processing applications. However, the estimation of channel SI requires *a priori* knowledge about the channel impulse response (CIR) and/or the channel statistics. In practice, both the CIR and the channel statistics may vary with time. Hence, the classical approach to channel estimation often requires the use of training sequences and adaptive filters [21], [22]. However, alternate channel estimation strategies can be realized by the use of Markov models. Since Markov models replace the CIR by a state diagram, the task of channel estimation simplifies into a simple trellis based search process that can be implemented using the well known sequence estimation and/or symbol-by-symbol estimation algorithms.

The aim of this chapter is to investigate and develop different channel estimation algorithms for use over a Markov fading channel. Section (4.1) presents a trellis based interpretation of a Markov fading channel. Section (4.2) describes the derivation of the different channel estimation algorithms. Section (4.3) presents a theoretical framework for analyzing the performance of these algorithms, and Section (4.4) substantiates these theoretical inferences using results from computer simulations.

4.1 Fading Channel Model

Consider a hidden Markov model (HMM) of a slow, flat fading Rayleigh channel. Let $\mathbf{y}_1^T$ be the transmitted binary PSK sequence over this Markov fading channel. Then, assuming coherent detection, the received sequence, $\mathbf{r}_1^T$, is given by

$$r_i = a_i y_i + n_i \tag{4.1}$$

where r_i is the i-th received sample, a_i is the corresponding fading amplitude (output symbol

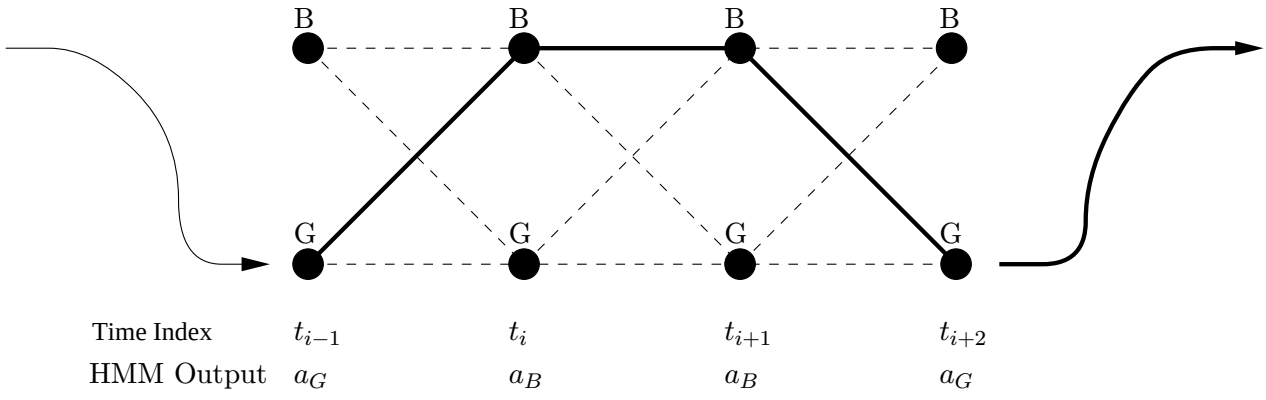

Figure 4.1: Trellis structure for a 2-state Markov fading channel.

of the HMM), and n_i is a zero-mean, additive, white Gaussian noise sample with variance σ^2. Recall that the time expansion of a HMM's state diagram yields a trellis. Hence, the sequence of fading amplitudes trace a path through the trellis defined by the HMM. A transmitted sequence can be thought to traverse the same path as the fading amplitudes. For instance, Figure (4.1) shows a possible sequence of fading amplitudes and the corresponding trellis path traversed by the transmitted sequence. The transmission of a sequence over a Markov fading channel is, therefore, akin to the convolutional and/or trellis encoding of a data sequence, where the encoding function is given by (4.1). Hence, much like convolutional and/or trellis decoding, the problem of channel estimation simplifies to estimating the path taken by the transmitted sequence.

4.2 Channel Estimation Algorithms

The aim of channel estimation is to estimate the output symbol sequence, $\mathbf{a}_1^T$, of the HMM, given the transmitted sequence $\mathbf{s}_1^T$ and the received sequence $\mathbf{r}_1^T$. Unlike classical channel estimation for the Rayleigh fading channel where optimum estimates are impossible to obtain, a trellis based solution for a Markov fading channel allows us to compute optimum channel estimates [83]. This optimum, maximum *a posteriori* (MAP), estimate is given by

$$\hat{\mathbf{a}}_1^T = \arg\max_{\mathbf{a}_1^T} \left[Pr\{\mathbf{a}_1^T, \mathbf{s}_1^T | \mathbf{r}_1^T\} \right] \tag{4.2}$$

where the maximization is over all the possible channel output symbol sequences. The *a posteriori* probability in (4.2) can be expressed as

$$Pr\{\mathbf{a}_1^T, \mathbf{s}_1^T | \mathbf{r}_1^T\} = \frac{Pr\{\mathbf{a}_1^T, \mathbf{s}_1^T, \mathbf{r}_1^T\}}{Pr\{\mathbf{r}_1^T\}} \tag{4.3}$$

and can be rewritten in terms of the *a priori* probabilities, i.e.,

$$Pr\{\mathbf{a}_1^T, \mathbf{s}_1^T | \mathbf{r}_1^T\} = \frac{Pr\{\mathbf{r}_1^T | \mathbf{a}_1^T, \mathbf{s}_1^T\} Pr\{\mathbf{a}_1^T, \mathbf{s}_1^T\}}{Pr\{\mathbf{r}_1^T\}} \tag{4.4}$$

However, since the transmitted sequence and the output symbol sequence of the HMM are independent, (4.4) can be written as

$$Pr\{\mathbf{a}_1^T, \mathbf{s}_1^T | \mathbf{r}_1^T\} = \frac{Pr\{\mathbf{r}_1^T | \mathbf{a}_1^T, \mathbf{s}_1^T\} Pr\{\mathbf{a}_1^T\} Pr\{\mathbf{s}_1^T\}}{Pr\{\mathbf{r}_1^T\}} \tag{4.5}$$

Since the probability $Pr\{\mathbf{r}_1^T\}$ is same for all the possible channel output symbol sequences, the MAP estimate in (4.2) can be modified as

$$\hat{\mathbf{a}}_1^T = \arg\max_{\mathbf{a}_1^T} \left[Pr\{\mathbf{r}_1^T | \mathbf{a}_1^T, \mathbf{s}_1^T\} Pr\{\mathbf{a}_1^T\} Pr\{\mathbf{s}_1^T\} \right] \tag{4.6}$$

Furthermore, since the HMM is first-order, the argument of (4.6) can be expressed as

$$Pr\{\mathbf{r}_1^T | \mathbf{a}_1^T, \mathbf{s}_1^T\} Pr\{\mathbf{a}_1^T\} Pr\{\mathbf{s}_1^T\} = \prod_{k=1}^{T} Pr\{r_k | a_k, s_k\} Pr\{a_k | a_{k-1}\} Pr\{a_0\} Pr\{s_k\} \tag{4.7}$$

where $Pr\{a_k | a_{k-1}\}$ is the transition probability between any two states (recall that there exists a one-to-one correspondence between the states and the alphabet of the HMM), $Pr\{a_0\}$ is the initial state probability, and $Pr\{s_k\}$ is the probability of transmitting the k-th symbol from the modulation alphabet. Assume that $Pr\{s_k\} = 1/M$ for M-ary modulation. Also assume that the HMM has N states. Then, using Baye's theorem, the RHS of (4.7) can be written as

$$\prod_{k=1}^{T} \left[\sum_{j=1}^{N} Pr\{r_k | a_k(j), s_k\} Pr\{a_k(j)\} \right] Pr\{a_k | a_{k-1}\} Pr\{a_0\} \frac{1}{M} \tag{4.8}$$

where $Pr\{a_k(j)\}$ is the observation probability of the k-th output symbol in the j-th state of the HMM. Since the underlying noise process is zero-mean Gaussian with variance σ^2, (4.8) can be rewritten as

$$\frac{1}{M^T} \prod_{k=1}^{T} \left[\sum_{j=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ \frac{-1}{2\sigma^2} (r_k - a_k(j) s_k)^2 \right\} Pr\{a_k(j)\} \right] Pr\{a_k | a_{k-1}\} Pr\{a_0\} \tag{4.9}$$

Either sequence estimation algorithms such as the Viterbi algorithm or symbol-by-symbol estimation algorithms such as the forward-backward algorithm can be used to implement (4.9) in order to estimate the optimum HMM (channel) output symbol sequence.

4.2.1 Sequence Estimation

Sequence estimaion algorithms such as the Viterbi algorithm (VA) can be used to derive an estimate, $\hat{\mathbf{a}}_1^T$, of the HMM's output symbol sequence. Define the forward variable $\delta_\tau(n)$ as

$$\delta_\tau(n) = Pr\{r_1, r_2, \dots, r_\tau, a_n, s_1, s_2, \dots, s_\tau\} \tag{4.10}$$

where $\delta_\tau(n)$ is the joint probability of receiving $\mathbf{r}_1^\tau$ and transmitting $\mathbf{s}_1^\tau$ when the channel is in state $n \in \{1, 2, \dots, N\}$ at time τ. These forward probabilities are initialized as

$$\delta_1(n) = \pi_n Pr\{r_1|a_n, s_1\} \tag{4.11}$$

where π_n is the initial state probability for the n-th HMM state, and $Pr\{r_1|a_n, s_1\}$ is the conditional probability of the first received symbol.

Likewise, $\delta_{\tau-1}(n)$ can be defined as the joint probability of receiving $\mathbf{r}_1^{\tau-1}$ and transmitting $\mathbf{s}_1^{\tau-1}$ when the HMM is in state n at time $\tau - 1$. Then, the product $\delta_{\tau-1}(n)Pr\{m|n\}$ is the joint probability of receiving $\mathbf{r}_1^{\tau-1}$ and transmitting $\mathbf{s}_1^{\tau-1}$ when the HMM is in state m at time τ. Choosing the largest product from the N possible states at time $\tau - 1$ gives a measure of the probability of being in state m at time τ with all the accompanying previous partial observations. Therefore, $\delta_\tau(n)$ can be computed recursively as

$$\delta_\tau(m) = \max_{1 \le n \le N} [\delta_{\tau-1}(n)Pr\{m|n\}]Pr\{r_\tau|a_m, s_\tau\} \qquad \tau \in \{2, 3, \dots, T\} \tag{4.12}$$

where $m \in \{1, 2, \dots, N\}$. Moreover, define the past state index, $\psi_\tau(n)$, as the state n that maximizes (4.12). The past state index is initialized as

$$\psi_1(n) = 0 \qquad n \in \{1, 2, \dots, N\} \tag{4.13}$$

and can be computed recursively as

$$\psi_\tau(m) = \arg\max_{1 \le n \le N} [\delta_{\tau-1}(n)Pr\{m|n\}] \tag{4.14}$$

Let s_T^* be the terminal state with the largest probability $\delta_T(n)$, i.e.,

$$s_T^* = \arg\max_{1 \le n \le N} [\delta_T(n)] \tag{4.15}$$

Then for $\tau \in \{T - 1, T - 2, \dots, 1\}$, the estimated state sequence, s_τ^*, can be obtained as

$$s_\tau^* = \psi_{\tau+1}(s_{\tau+1}^*) \tag{4.16}$$

Once the sequence of the most likely states is determined, the estimated output symbol sequence, $\hat{\mathbf{a}}_1^T$, can be found by using the one-to-one correspondence between the states and the alphabet of the HMM.

4.2.2 Symbol-by-Symbol Estimation

Symbol-by-symbol estimation algorithms such as the forward-backward algorithm (FBA) offer an alternate solution for estimating the output symbol sequence, $\hat{\mathbf{a}}_1^T$, of the HMM. Define the forward variable $\alpha_\tau(n)$ in a manner similar to (4.10), i.e.,

$$\alpha_\tau(n) = Pr\{r_1, r_2, \dots, r_\tau, a_n, s_1, s_2, \dots, s_\tau\} \tag{4.17}$$

where $n \in \{1, 2, \dots, N\}$. The forward variable is initialized as

$$\alpha_1(n) = \pi_n Pr\{r_1 | a_n, s_1\} \tag{4.18}$$

Since $\alpha_\tau(n)$ is the joint probability of receiving $\mathbf{r}_1^\tau$ and transmitting $\mathbf{s}_1^\tau$ when the HMM is in state n at time τ, the product $\alpha_\tau(n)Pr\{m|n\}$ is the probability of receiving $\mathbf{r}_1^\tau$ and transmitting $\mathbf{s}_1^\tau$ when the HMM is in state m at time $\tau + 1$. Summing this product over all the N possible states at time τ gives the total probability of being in state m at time $\tau+1$ with all the accompanying previous partial observations. Therefore, $\alpha_{\tau+1}(n)$ can be obtained by augmenting multiplicatively this summed quantity with the conditional probability $Pr\{r_\tau | a_n, s_\tau\}$, i.e.,

$$\alpha_{\tau+1}(m) = \left[\sum_{n=1}^{N} \alpha_\tau(n) Pr\{m|n\}\right] Pr\{r_\tau | a_m, s_\tau\} \qquad \tau \in \{1, 2, \dots, T-1\} \tag{4.19}$$

where $m \in \{1, 2, \dots, N\}$. Likewise, the backward variable $\beta_\tau(n)$ is defined as

$$\beta_\tau(n) = Pr\{r_{\tau+1}, r_{\tau+2}, \dots, r_t, a_n, s_{\tau+1}, s_{\tau+2}, \dots, s_t\} \qquad n \in \{1, 2, \dots, N\} \tag{4.20}$$

and is the probability of receiving $\mathbf{r}_{\tau+1}^T$ and transmitting $\mathbf{s}_{\tau+1}^T$ when the HMM is in state n at time τ. The backward variable is initialized as

$$\beta_T(n) = 1 \qquad \forall\ n \tag{4.21}$$

In order to have been in state n at time τ, the FBA needs to account for a transition to every one of the N possible states at time $\tau + 1$, account for the received and transmitted symbols in each of these N states, and then account for the rest of the observation sequence. Therefore, the backward variables can be computed recursively as

$$\beta_\tau(n) = \sum_{m=1}^{N} Pr\{m|n\} Pr\{r_{\tau+1} | a_m, s_{\tau+1}\} \qquad \tau \in \{T-1, T-2, \dots, 1\} \tag{4.22}$$

Therefore, the probability of being in state n at time τ, given the entire received sequence and the transmitted sequence is given by

$$\gamma_\tau(n) = \alpha_\tau(n)\beta_\tau(n) \qquad \tau \in \{1, 2, \dots, T\} \tag{4.23}$$

Using (4.23), the individually most likely state at time τ is given by

$$s_\tau = \arg\max_{1\leq n\leq N}[\gamma_\tau(n)] \qquad \tau \in \{1, 2, \ldots, T\} \tag{4.24}$$

Once again, the estimated output symbol sequence, $\hat{\mathbf{a}}_1^T$, can be found by using the one-to-one correspondence between the states and the alphabet of the HMM.

4.3 Theoretical Performance Analysis

We shall use the Gilbert HMM to analyze the performance of the MAP channel estimator. Recall that the Gilbert model has two states, *good* and *bad*, denoted by G and B. Let $P_{G,G}$ be the virtual probability of transition from the good state to the good state. Likewise, let $P_{B,B}$ be the virtual probability of transition from the bad state to the bad state. Let $P_{G,B}$ and $P_{B,G}$ be the real transition probabilities. Assume that $P_{G,G} \geq P_{B,B}$, $P_{B,G} \geq P_{G,B}$, and $P_{G,G} >> P_{G,B}$[1]. Let a_G and a_B be the fading amplitudes (output symbols) associated with the two states of the HMM. Then, using (4.6) and (4.9), the channel estimator will choose the good state for the i-th received symbol iff

$$\frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-1}{2\sigma^2}(r_i - a_G s_i)^2\right\} Pr\{G\} > \frac{1}{\sigma\sqrt{2\pi}}\exp\left\{\frac{-1}{2\sigma^2}(r_i - a_B s_i)^2\right\} Pr\{B\} \tag{4.25}$$

where using (4.7), $Pr\{G\}$ and $Pr\{B\}$[2] are given as

$$Pr\{G\} = Pr\{G|s_{i-1}\}\prod_{k=1}^{i-1} Pr\{s_k|s_{k-1}\}Pr\{s_0\} \tag{4.26}$$

and

$$Pr\{B\} = Pr\{B|s_{i-1}\}\prod_{k=1}^{i-1} Pr\{s_k|s_{k-1}\}Pr\{s_0\} \tag{4.27}$$

where s_{i-1} is the state occupied by the HMM at time $i-1$. Using the results from (4.26) and (4.27), and discarding the common terms on either side of the inequality in (4.25) gives

$$\exp\left\{\frac{-1}{2\sigma^2}(r_i - a_G s_i)^2\right\} Pr\{G|s_{i-1}\} > \exp\left\{\frac{-1}{2\sigma^2}(r_i - a_B s_i)^2\right\} Pr\{B|s_{i-1}\} \tag{4.28}$$

[1] These assumptions are valid in light of the results presented in Section (3.5.1).

[2] Recall that there exists a one-to-one correspodence between the states and the alphabet of the HMM.

Rearranging (4.28) after taking its natural logarithm gives the decision criterion for choosing the good state

$$(r_i - a_G s_i)^2 - (r_i - a_B s_i)^2 < 2\sigma^2 ln \left[\frac{Pr\{G|s_{i-1}\}}{Pr\{B|s_{i-1}\}}\right] \tag{4.29}$$

The performance of the channel estimator can be characterized by considering two extreme scenarios, viz., $a_G \approx a_B$, and $a_G >> a_B$. In both cases, we assume that the receiver has perfect knowledge of the transmitted sequence. Furthermore, let the i-th transmitted symbol be $+1$, and let the i-th channel state be the bad state. Note that with this latter assumption the channel estimator will make an incorrect decision if (4.29) holds true.

First, consider the case when $a_G = a_B^+$, i.e., the output symbol associated with the good state is slightly larger than the output symbol associated with the bad state. Then,

$$(r_i - a_G s_i)^2 - (r_i - a_B s_i)^2 = \delta \tag{4.30}$$

where δ is a small positive number. When σ^2 is much smaller then δ, then the RHS of (4.29) will either be a positive number less than δ if $s_{i-1} = G$, or a negative number less than δ if $s_{i-1} = B$. In either case, the channel estimator will make the correct decision. Conversely, when σ^2 is much larger than δ, then the RHS of (4.29) will either be a positive number larger than δ if $s_{i-1} = G$, or a negative number much smaller than δ if $s_{i-1} = B$. Hence, the channel estimator will make an error whenever $s_{i-1} = G$.

Second, consider the case when $a_G >> a_B$, i.e, the Euclidean distance between the output symbols of the HMM is large. Then,

$$(r_i - a_G s_i)^2 - (r_i - a_B s_i)^2 = \Delta \tag{4.31}$$

where Δ is a large positive number. Once again, when σ^2 is much smaller than Δ, the channel estimator makes correct decisions regardless of the state occupied by the HMM at time $i-1$. Likewise, when σ^2 is large and $s_{i-1} = B$, the channel estimator makes the correct decision. However, when σ^2 is large and $s_{i-1} = G$, then the RHS of (4.29) may or may not be larger than Δ. Therefore, the channel estimator may or may not make an incorrect decision.

It can, therefore, be inferred from the above analysis that the performance of the channel estimator is sensitive to the noise variance, the i-th step transition probability, and the Euclidean distance between the output symbols of the HMM. This sensitivity is especially large when the Euclidean distance is small, when the noise variance is large, and when the HMM experiences real transitions. Moreover, the difference in magnitude of the virtual transition probabilities will bias the threshold—the RHS of (4.29)—such that a large percentage of incorrect estimates will include the state with the largest virtual transition probability. Furthermore, inaccurate estimates of the transmitted sequence (ignored in the above analysis) will also contribute towards the performance degradation of the channel estimator.

4.4 Simulation Results

Several computer simulations were performed to characterize the performance of the channel estimator. In particular, these simulations were used to analyze the sensitivity of the channel estimator to the Euclidean distance between the output symbols of the HMM (or to the granularity of the HMM), high and low fidelity estimates of the transmitted sequence, variations in the noise variance, and the normalized fading rate of the channel.

All simulations use a random binary PSK modulated transmitted sequence, a block length of 600 samples, and a stopping criterion of 1000 errors. The simulations use a 2-state model to verify the theoretical inferences made in Section (4.3), and then proceeds with 32-state and 64-state models to characterize the performance of the channel estimator over typical fading environments. Unless otherwise specified, all HMMs used in this study correspond to a fading channel with a normalized fading rate, $f_d T_s = 0.01$.

Two different metrics are used to benchmark the performance of the channel estimator. These metrics include the channel state estimation error rate (SER), and the confusion matrix (CM). The SER is defined as the ratio of incorrect decisions to the total number of decisions made by the channel estimator. For a N-state HMM, the channel estimator can make one of N decisions at every time instant, i.e., it can choose the correct HMM state, or it can choose one out of $N-1$ incorrect states. Therefore, a $N \times N$ matrix whose rows represent the actual HMM state, and the columns represent the estimated state can be used to illustrate the distribution of correct and incorrect decisions made by the channel estimator. This $N \times N$ matrix is called the confusion matrix (CM).

4.4.1 Sensitivity to Model Granularity

The 2-state HMM developed in Section (3.5.1) is used to verify the theoretical inferences made in Section (4.3). We assume that the receiver has perfect knowledge of the transmitted sequence and the noise variance. The Euclidean distance between the two states of the HMM is varied by changing the output symbol associated with the good state from -4 dB to 10 dB, while keeping the output symbol of the bad state constant at -10 dB (recall that the lowest value of the good state must be greater than the threshold, -5 dB, used to construct the HMM). Figure (4.2) shows the SER profile of the Viterbi channel estimator versus the output symbol, SNR_G, of the good state. Superimposed on the SER profile are the CM probabilities denoted by $Pr(s_e = i | s = j)$, where s_e is the estimated state, s is the actual state, and $i, j \in \{G, B\}$. It is evident from the SER profile that the performance of the channel estimator improves as the separation or the Euclidean distance between the output symbols increases. Note that the SER plot is bounded on either side by the profiles of the incorrect CM probabilities, i.e., the probabilities of over and under estimation. It is also interesting to note that the CM probabilities for over estimation are larger than their counterparts for under estimation regardless of the Euclidean distance. This inherent bias

towards over estimation arises from the fact that the virtual probability $P_{G,G}$ is larger than the virtual probability $P_{B,B}$.

We now consider the performance of the channel estimator over more practical 32-state and 64-state HMMs. Figure (4.3) shows the SER versus E_b/N_o plots for both the Viterbi and the forward-backward channel estimators. As expected, the channel estimators perform poorly over the 64-state HMM, where the Euclidean distance between the states is much smaller than the distance between the states of the 32-state HMM. This difference in performance can be explained in terms of the CM probabilities. For instance, consider the sub-set of CM probabilities $Pr(s_e = i|s = j, a_j \approx 1)$ where the state j corresponds to a fading amplitude, a, of approximately one (corresponds to level 4 in the 32-state model, and to level 7 in the 64-state model). Figure (4.4) shows the distribution of these CM probabilities at different signal-to-noise ratios for the Viterbi channel estimator. Both the over and under estimation probabilities decrease with increasing signal-to-noise ratio. However, this decrease is more pronounced in case of the 32-state HMM, than for the 64-state HMM. For instance, at an E_b/N_o of 20 dB, the total probability of over and under estimation is negligible in case of the 32-state HMM, but amounts to 1/10-th of all decisions in case of the 64-state HMM. However, for both the 32-state HMM and the 64-state HMM, the CM probabilities are distributed over the immediate neighboring states, and not the entire state space of the HMM. Therefore, increasing the number of states increases the probability of incorrect decisions, but it decreases the magnitude of the average estimation error[3], since adjacent states are separated by smaller Euclidean distances. It is important to realize that while the state estimation error rate (SER) may be high, the actual estimation error may be small. This is especially true for HMMs with a large number of states. Hence, SER can, *per se*, be a misleading measure of the performance of the channel estimator, and must, therefore, be used in conjunction with the CM probabilities.

4.4.2 Influence of Transmitted Sequence and Noise Variance

Figure (4.5) shows the SER versus E_b/N_o plots over a 32-state HMM for both the Viterbi and the forward-backward channel estimator. The plots depict the performance of the two estimators for instances when the receiver has perfect knowledge of the transmitted sequence and the noise variance, and for instances when the receiver has to estimate both the transmitted sequence and the noise variance[4]. It is evident that imperfect cognizance of the transmitted sequence and the noise variance can significantly degrade the performance of the two channel estimators. However, the forward-backward channel estimator performs slightly better than its Viterbi counterpart under these hostile conditions (imperfect estimates). Figure (4.6) shows the distribution of CM probabilities $Pr(s_e = i|s = j, a_j \approx 1)$ for the Viterbi channel

[3]The estimation error is defined as the absolute difference between the estimated fading amplitude and the desired (actual) fading amplitude.

[4]The transmitted sequence is estimated by making hard decisions on the received sequence. The noise variance is estimated by computing the sample variance of the received frame.

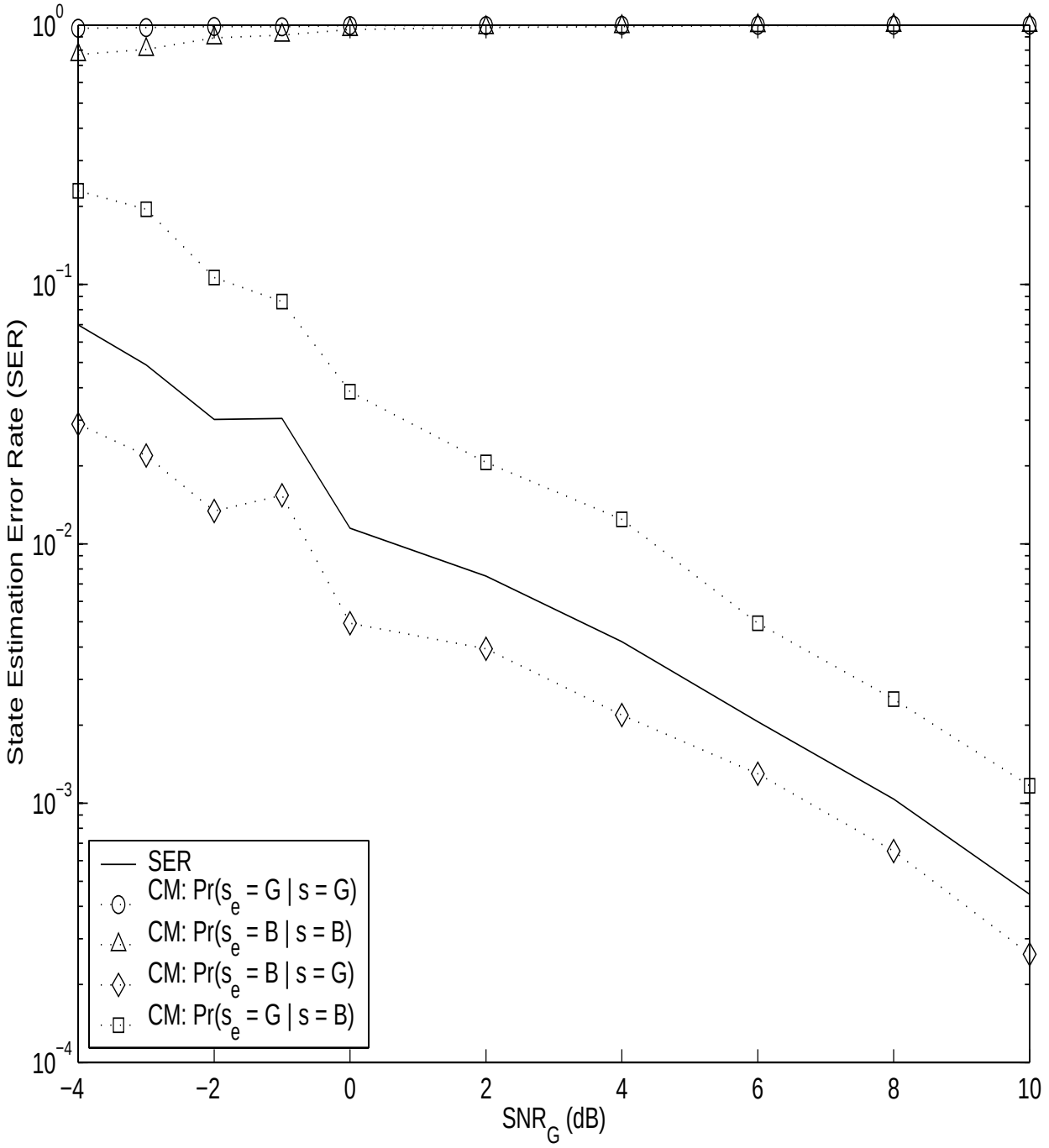

Figure 4.2: Performance of the Viterbi estimator over a Gilbert HMM. The graph shows the channel state estimation error rate and confusion matrix probabilities versus the output symbol of the good state. Assume that $SNR_B = -10$ dB, and that the receiver has perfect knowledge of the transmitted sequence.

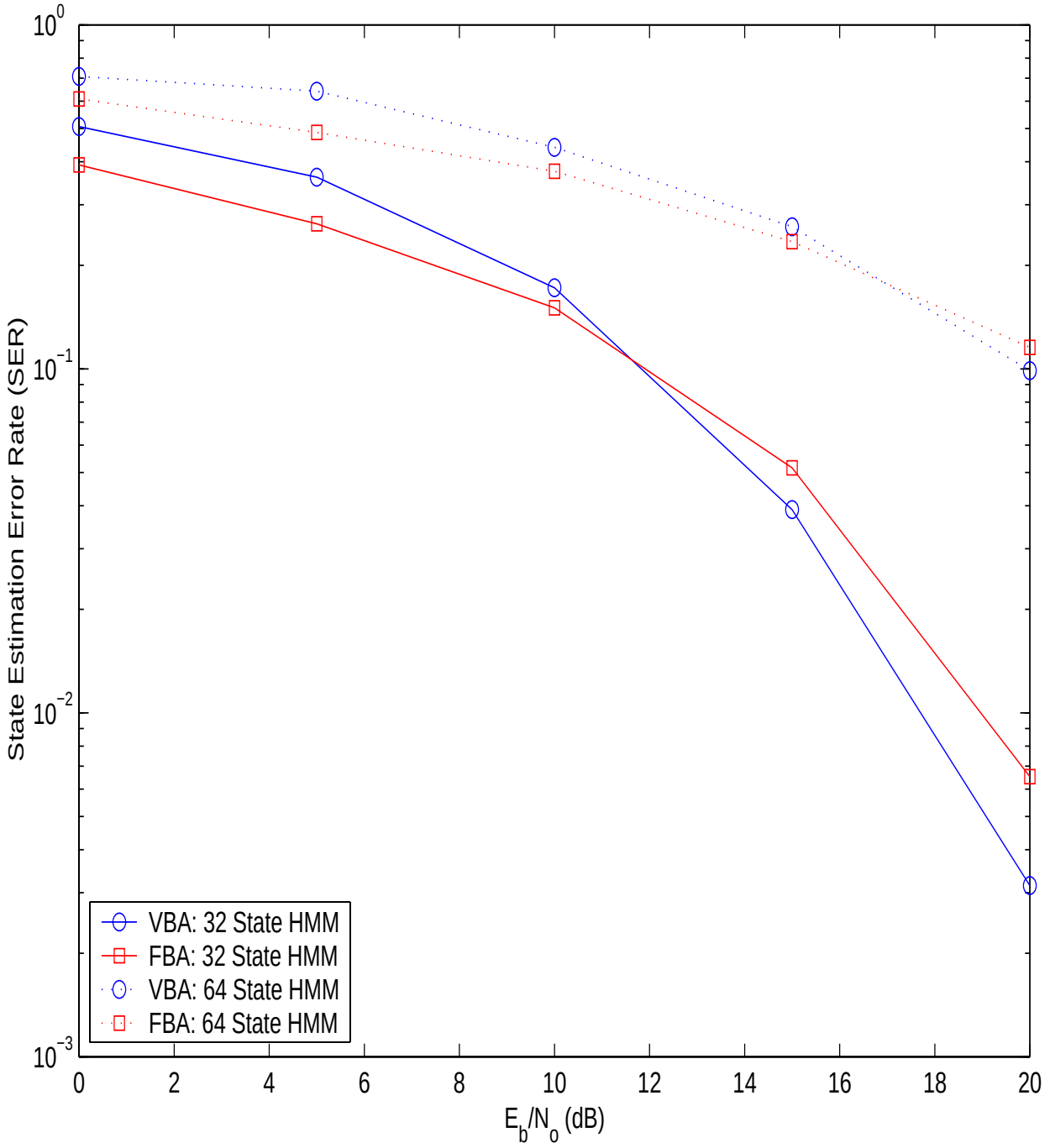

Figure 4.3: Performance of the Viterbi and the forward-backward channel estimators over a 32-state and a 64-state HMM with normalized fading rate $f_d T_s = 0.01$. The graph shows the channel state estimation error rate (SER) versus E_b/N_o. Assume that the receiver has perfect knowledge of the transmitted sequence and the noise variance.

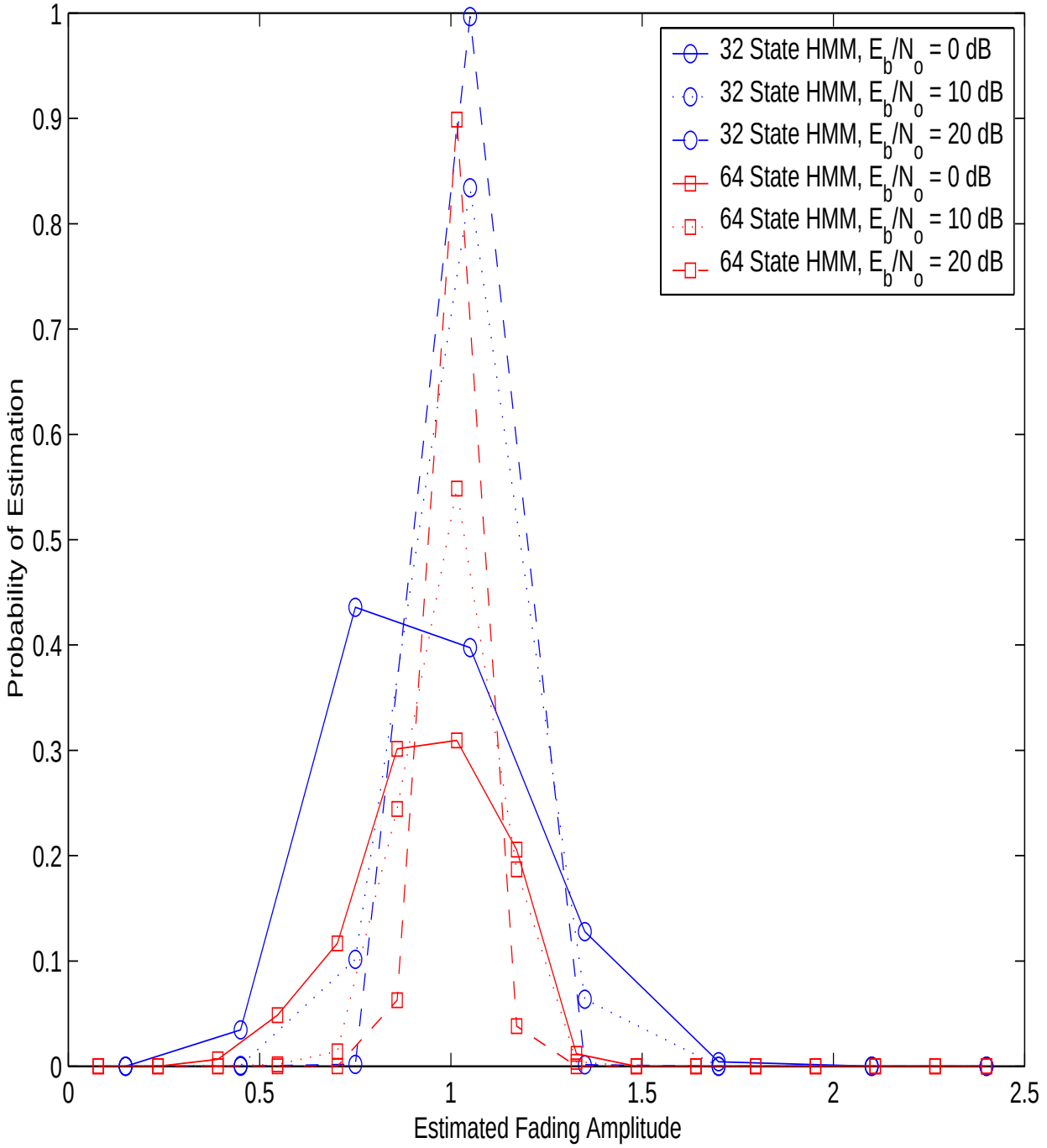

Figure 4.4: Distribution of CM probabilities for the Viterbi channel estimator as a function of the Euclidean distance between the HMM states. Desired fading amplitude is approximately one.

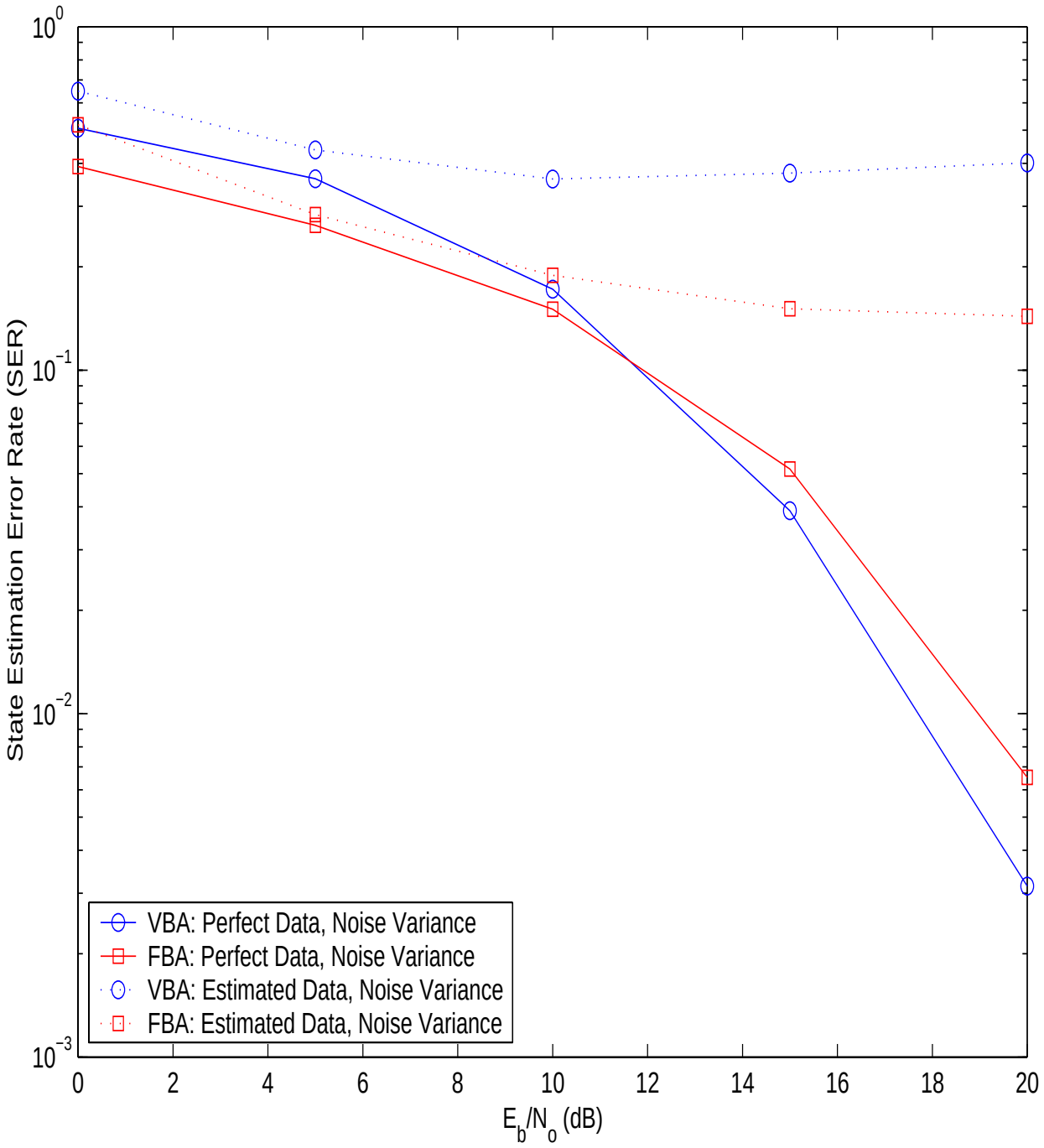

Figure 4.5: Performance of the Viterbi and the forward-backward channel estimators over a 32-state HMM with normalized fading rate $f_d T_s = 0.01$. The plots depict the sensitivity of the channel estimators to the estimates of the transmitted sequence and the noise variance.

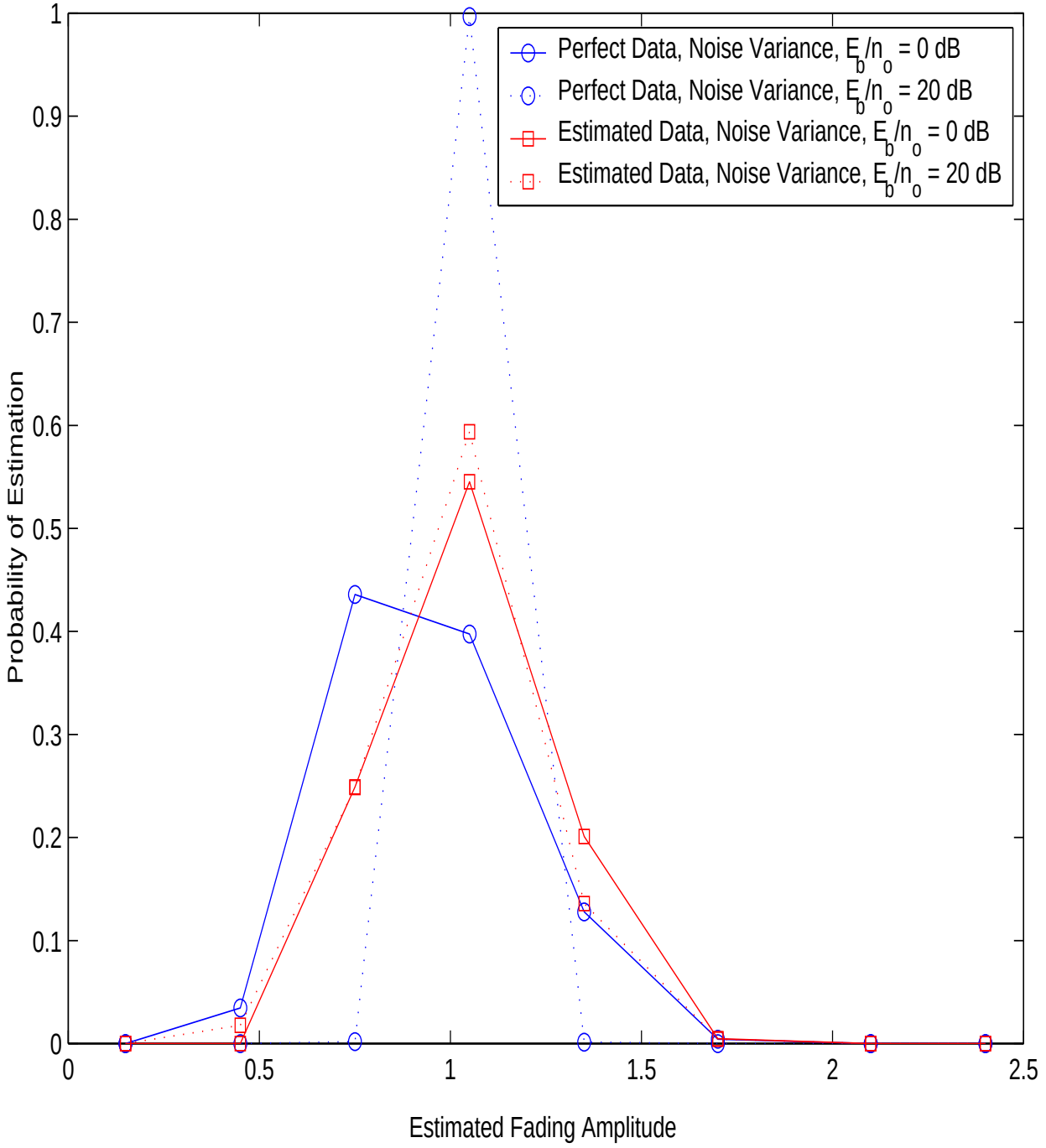

Figure 4.6: Distribution of CM probabilities for the Viterbi channel estimator as a function of the fidelity of the estimates of the transmitted sequence and the noise variance. Desired fading amplitude is approximately one.

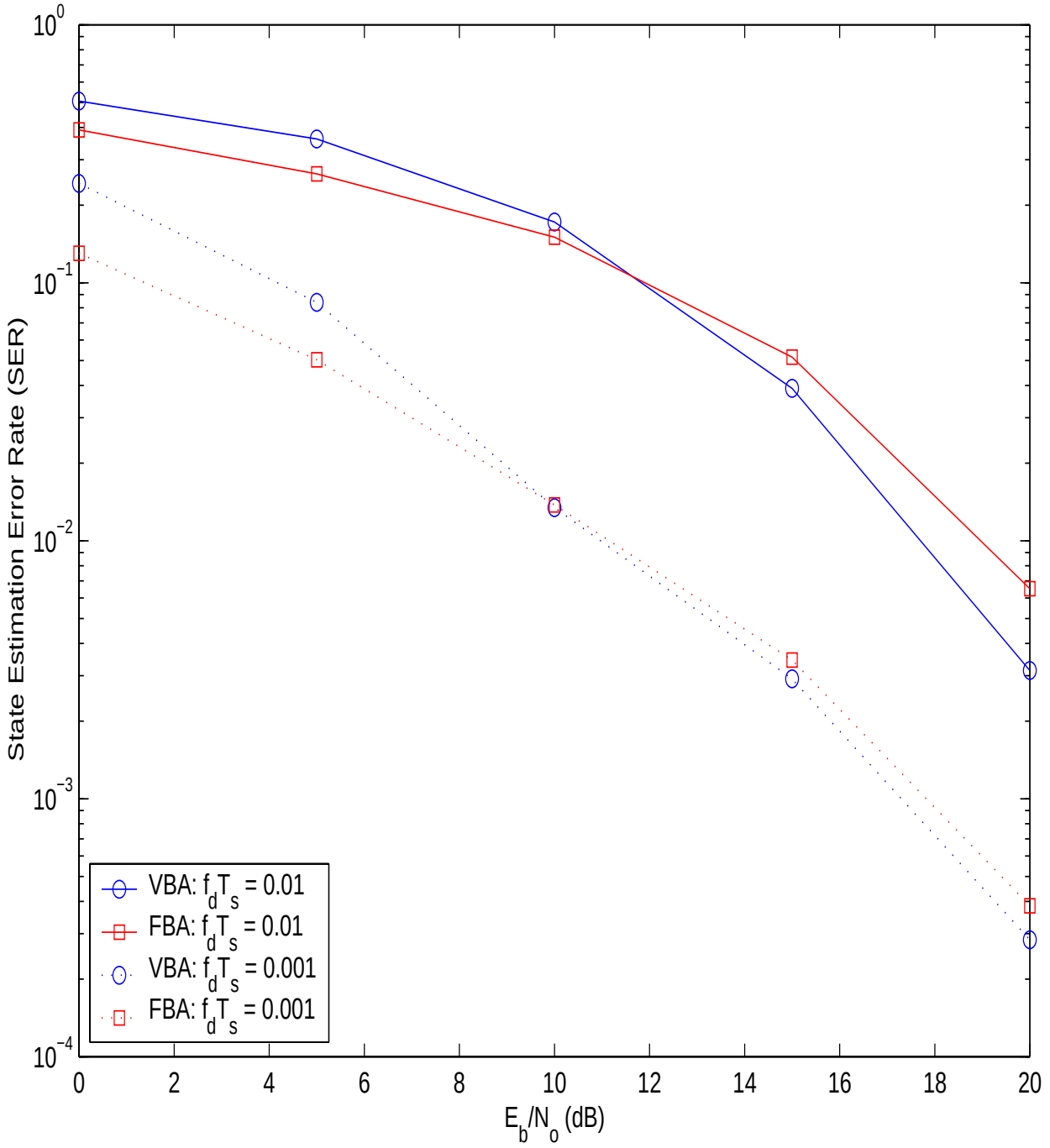

Figure 4.7: Performance of the Viterbi and the forward-backward channel estimators over two 32-state HMMs with normalized fading rates $f_dT_s = 0.01$ and $f_dT_s = 0.001$. The plots depict the sensitivity of the channel estimators to the fading rate or to the frequency of real transitions in the Markov fading channel. We assume that the receiver has perfect knowledge of the transmitted sequence and the noise variance.

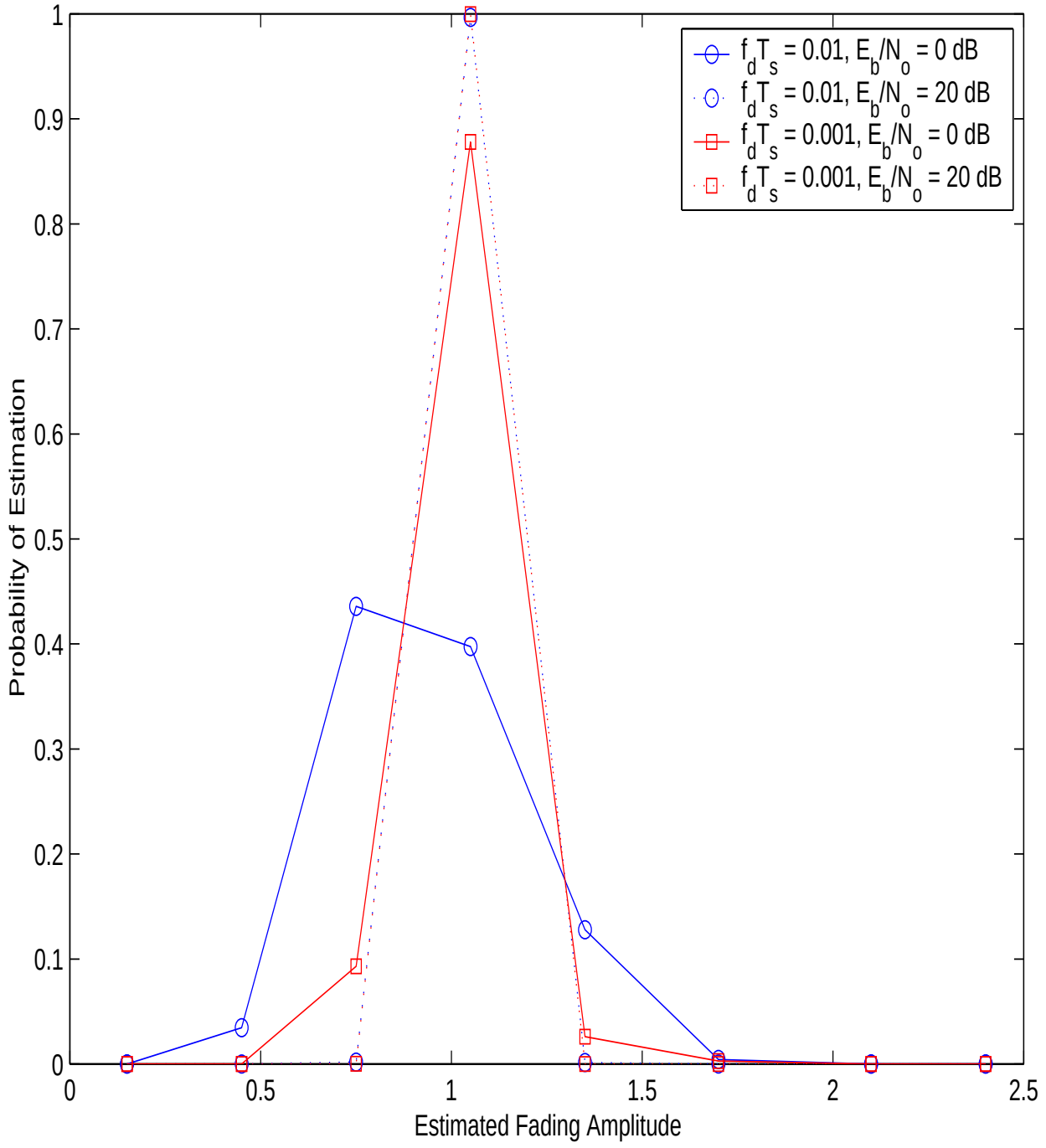

Figure 4.8: Distribution of CM probabilities for the Viterbi channel estimator as a function of the fading rate. Desired fading amplitude is approximately one.

estimator at different signal-to-noise ratios. It is interesting to note that a 20 dB increase in E_b/N_o causes only a slight decrease in the probability of over estimation for the case with imperfect cognizance, while a similar increase in E_b/N_o causes a significant decrease in both the over and under estimation probabilities for the case with perfect cognizance.

4.4.3 Effect of Fading Rate

Figure (4.7) shows the performance of the Viterbi and the forward-backward channel estimators over two 32-state HMMs with normalized fading rates $f_dT_s = 0.01$ and $f_dT_s = 0.001$. For both cases, we assume that the receiver has perfect knowledge of the transmitted sequence and the noise variance. Once again, as expected the two channel estimators perform better when the fading rate is low—when the frequency of real transitions is small—than when the fading rate is high. Figure (4.8) shows the distribution of CM probabilities $Pr(s_e = i|s = j, a_j \approx 1)$ for the Viterbi channel estimator. The improved SER performance at low fading rates can be attributed to the significant reduction in the probability of over and under estimation.

4.5 Chapter Summary

The ability to model a fading channel using Markov models facilitate a simple trellis based solution to the otherwise complicated task of channel estimation. A data sequence transmitted over a Markov fading channel traverses a unique path through the trellis defined by the Markov model. Thus, channel estimation simplifies to a problem of estimating either the most likely state sequence or the most likely individual states of the Markov model that correspond to the original path taken by the transmitted sequence.

This chapter presented a detailed derivation of the maximum *a posteriori* (MAP) channel estimation algorithm, and described the application of both the Viterbi and the forward-backward algorithms to channel estimation over a Markov fading channel. The chapter provided a simple theoretical analysis of the aforementioned MAP algorithm, and substantiated the theoretical inferences with results from computer simulations. The channel estimator was shown to be sensitive to the estimates of the noise variance and the transmitted sequence, the fading rate of the channel, and to the Euclidean distance between the states of the Markov model. Future work should include a more mathematically robust analysis of the MAP channel estimation algorithm. It should streamline both the Viterbi and the forward-backward algorithms, perhaps by implementing them in the log-domain. Future studies should also study the affect of phase variations, and should include channel estimation over a variety of Markov fading channels, such as fast fading channels, frequency-selective fading channels, and fading channels where the received envelope has non-Rayleigh distributions.

Chapter 5

Iterative Decoding

Recent studies have shown that turbo codes can achieve remarkable bit error rate performance over flat fading channels [84], [85], [86]. However, many of these published studies assume perfect knowledge of the channel side information, viz., the fading amplitude, the carrier phase, and the noise variance. However, in practice, these channel parameters need to be estimated at the receiver. Typically, the fading amplitude and the carrier phase are jointly estimated as the complex channel gain using either pilot symbol assisted modulation (PSAM) or pilot tone assisted modulation (PTAM) techniques [21], [22]. Likewise, the noise variance is usually estimated by computing the sample variance of the received sequence in a decision-directed mode by using the estimates of the complex channel gain and the output of the turbo decoder [23]. In either case, the ability to reliably estimate the fading amplitude (or the complex channel gain) is of paramount importance. However, unlike classical adaptive filtering techniques that are used in conjunction with either PSAM or PTAM strategies, trellis based solutions using Markov fading models allow us to compute optimum channel (fading amplitude) estimates using well known techniques such as the Viterbi algorithm and/or the forward-backward algorithm. Furthermore, the use of Markov models facilitates a unified trellis based framework for the receiver, where operations such as demodulation, detection, equalization, and channel decoding can either be defined jointly on a hyper-trellis, or separately on individual trellises.

The aim of this chapter is to investigate iterative decoding strategies that employ trellis based channel estimators over Markov fading channels. Section (5.1) describes the channel model, and the channel reliability metric. Section (5.2) presents a modified version of the channel reliability metric that uses soft information about the channel states. Section (5.3) describes different iterative decoding and channel estimation strategies, and Section (5.4) benchmarks these strategies with results from computer simulations.

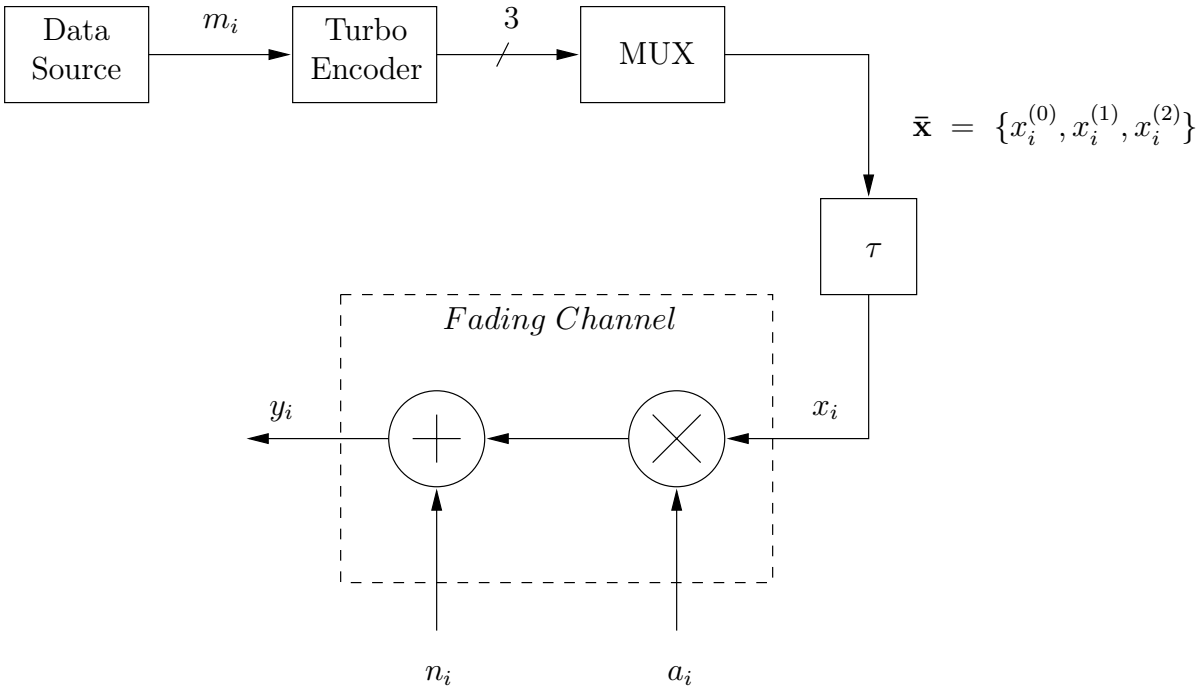

Figure 5.1: A turbo encoded communication system.

5.1 Channel Model and Channel Reliability

Consider the communication system shown is Figure (5.1). An information symbol, m_i, is encoded using a rate $r = 1/3$ turbo encoder. The outputs of the turbo encoder are multiplexed to form the codeword $\bar{\mathbf{x}} = \{x_i^{(0)}, x_i^{(1)}, x_i^{(2)}\}$. This codeword sequence is channel interleaved, binary PSK modulated, and transmitted over a slow, flat fading, Rayleigh channel. Assuming coherent detection

$$y_i = a_i(2x_i - 1) + n_i \tag{5.1}$$

where $x_i \in \{0, 1\}$ is the transmitted symbol, y_i is the received symbol, a_i is the fading amplitude, and n_i is a zero mean, white, Gaussian noise sample with variance σ^2. Using (2.53), the channel reliability metric for the received symbol, y_i, is given as

$$R_i = 2\frac{\hat{a}_i}{\hat{\sigma}^2}y_i \tag{5.2}$$

where $\hat{a}_i$ is the estimated fading amplitude, and $\hat{\sigma}^2$ is the estimated noise variance. Hence, the computation of the channel reliability metric necessitates the need for channel estimation

at the receiver. It has been shown that the lack of channel estimation or unreliable channel estimates can significantly degrade the performance of turbo codes [18], [19], [20]. Robust[1] channel estimation is, therefore, paramount in order to realize the large coding gains inherent in turbo coded schemes.

In order to compute the channel reliability metric, the fading amplitudes are estimated using the trellis based techniques described in Chapter (4), and the noise variance is estimated by computing the sample variance of the received sequence, i.e.,

$$\hat{\sigma}^2 = \frac{1}{L}\sum_{i=0}^{L-1}[y_i - \hat{a}_i(2\hat{x}_i - 1)]^2 \tag{5.3}$$

where $\hat{x}_i$ is the estimated transmitted symbol, and L is the length of the transmitted sequence. This framework allows us to perform channel estimation (fading amplitude and noise variance estimation) in a decision-directed mode where the outputs of the turbo decoder and the trellis estimator can be used iteratively to provide more reliable estimates of both the transmitted sequence and the fading ampltudes.

5.2 Weighted Channel Reliability

Recall that the Viterbi channel estimator yields the maximum likelihood (ML) sequence of the fading amplitudes, while the forward-backward channel estimator yields the maximum *a posteriori* (MAP) sequence of the fading amplitudes. Another important difference between the two algorithms is their ability to output soft information. The Viterbi[2] algorithm can only produce a hard output or a ML estimate of the fading amplitudes, without any accompanying measure of confidence. Conversely, the forward-backward algorithm can be modified to output soft information or the probability of state occupancy. As described in Chapter (4), the forward-backward algorithm estimates the fading amplitudes by selecting the states with the largest γ probabilities. Instead of making these hard decisions, the forward-backward algorithm can be configured to output these probabilities *per se*. Since the γ probability represents the total probability of being in a state given the entire received sequence, it provides a measure of confidence about observing a given fading amplitude at a given time instant. The computation of the channel reliability metric can, therefore, be modified to incorporate this soft information.

[1]It has been shown in [20] that turbo decoders can tolerate an SNR mismatch of -2 dB to 6 dB. Beyond this range, their performance degrades rapidly.

[2]The Viterbi algorithm refers to the one described in Chapter (4), and does not refer to other variations such as the SOVA algorithm.

Using (2.51) and (5.1), the channel reliability metric, R_i, can be defined as

$$R_i = \ln \left[\frac{Pr\{x_i = 1, a_i|\mathbf{y}\}}{Pr\{x_i = 0, a_i|\mathbf{y}\}} \right] \tag{5.4}$$

where x_i is the transmitted symbol, y_i is the received symbol, $\mathbf{y}$ is the received sequence, and a_i is the fading amplitude . Using Baye's theorem, the *a posteriori* probability $Pr\{x_i = b, a_i|\mathbf{y}\}$, $b \in \{0, 1\}$ can be written as

$$Pr\{x_i = b, a_i|\mathbf{y}\} = \frac{Pr\{y_i|x_i = b, a_i\}Pr\{x_i = b, a_i\}}{Pr\{\mathbf{y}\}} \tag{5.5}$$

where the *a priori* probability $Pr\{y_i|x_i, a_i\}$ is given by

$$Pr\{y_i|x_i, a_i\} = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{ \frac{-1}{2\sigma^2} \left[y_i - a_i(2x_i - 1) \right]^2 \right\} \tag{5.6}$$

Assume that both x_i and a_i are statistically independent, then (5.5) can be rewritten as

$$Pr\{x_i = b, a_i|\mathbf{y}\} = \frac{Pr\{y_i|x_i = b, a_i\}Pr\{x_i = b\}Pr\{a_i\}}{Pr\{\mathbf{y}\}} \tag{5.7}$$

Since a_i belongs to the N element set, $\mathbf{A}$, of fading amplitudes, (5.7) can be expressed as

$$Pr\{x_i = b, a_i|\mathbf{y}\} = \frac{\sum_{j=1}^{N} \left[Pr\{y_i|x_i = b, a(j)\}Pr\{a_i(j)\} \right] Pr\{x_i = b\}Pr\{a_i\}}{Pr\{\mathbf{y}\}} \tag{5.8}$$

where $a(j)$ is the output symbol (fading amplitude) associated with the j-th state of the HMM, and the probability $Pr\{a_i(j)\}$ equals the ij-th γ probability of occupying the j-th HMM state at time i. Hence, the *weighted* reliability metric can be defined as

$$\acute{R}_i = \ln \left[\frac{\sum_{j=1}^{N} Pr\{y_i|x_i = 1, a(j)\}\gamma_i(j)}{\sum_{j=1}^{N} Pr\{y_i|x_i = 0, a(j)\}\gamma_i(j)} \right] \tag{5.9}$$

5.3 Iterative Decoding and Channel Estimation

In principle, a Markov fading channel allows us to perform joint iterative decoding and channel estimation over a hyper-trellis that describes both the fading channel and the constituent RSC code. The hyper-trellis framework allows us to compute the optimal maximum

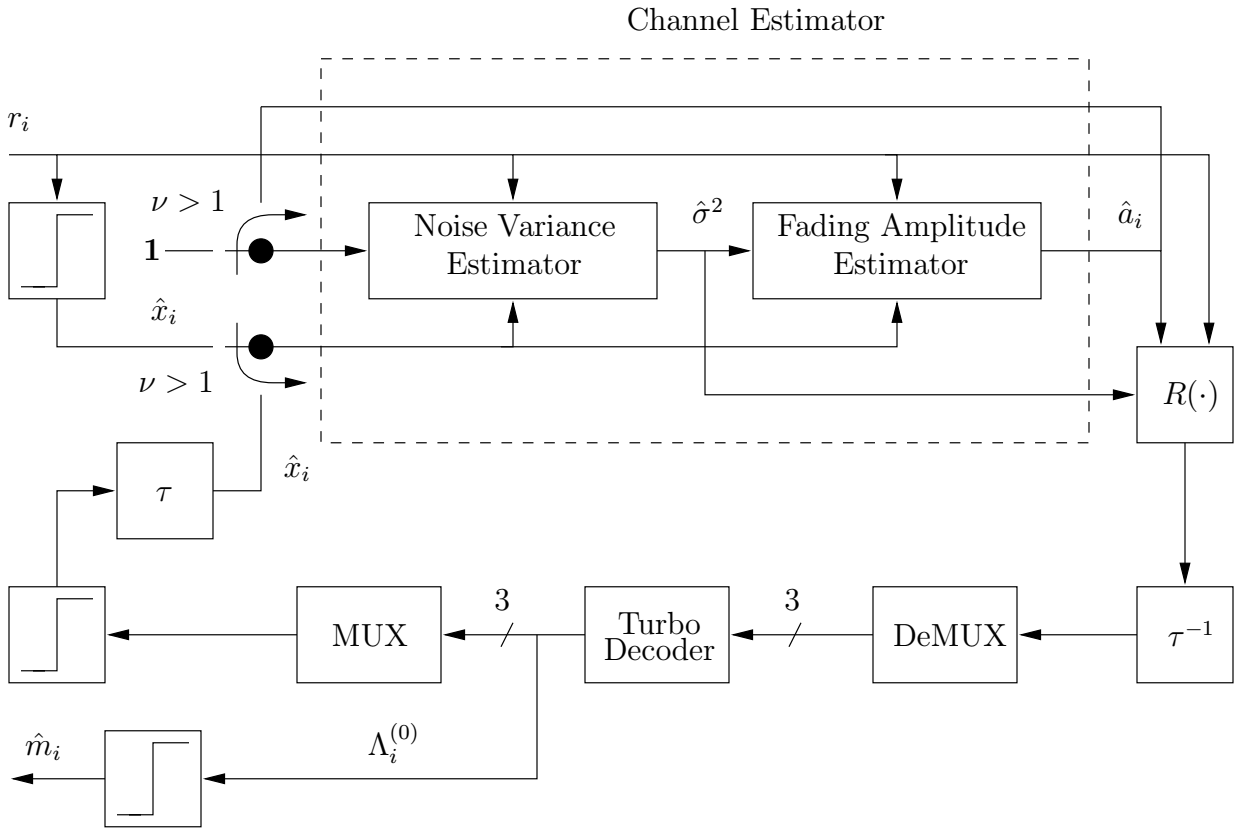

Figure 5.2: A turbo decoder with a trellis based channel estimator.

likelihood (ML) solution. However, since the complexity of trellis decoding increases exponentially with the number of states[3], this augmented trellis implementation yields an exorbitantly complex solution. Conversely, separating channel estimation and iterative decoding into two distinct tasks yields a non-optimal, but a more computationally efficient solution.

Figure (5.2) shows the block diagram of a turbo decoder, where channel estimation and iterative decoding are performed over separate trellises. The channel estimator accepts as inputs the received sequence, $\mathbf{r}$, the estimated transmitted sequence, $\hat{\mathbf{x}}$, and the estimated fading amplitude sequence, $\hat{\mathbf{a}}$. During the first pass through the channel estimator, the estimated transmitted sequence is obtained by thresholding the received sequence, and the estimated fading amplitudes are assumed to be unity. The channel estimator outputs the estimated noise variance, and the estimated fading amplitudes. The estimated fading amplitudes are fed back to the input of the channel estimator for use during subsequent iterations. The

[3]The number of states in the hyper-trellis equals the product of the number of states in the HMM and the number of states in the RSC code's trellis.

received sequence together with the estimated channel parameters are used to compute the channel reliability metrics. The sequence of reliability metrics is channel deinterleaved, demultiplexed, and fed to the turbo decoder. After η iterations of iterative decoding, the soft outptus of the turbo decoder can either be thresholded to yield an estimate of the information sequence, $\hat{\mathbf{m}}$, or can be multiplexed, thresholded, and channel interleaved to yield an estimate of the transmitted sequence. The estimated transmitted sequence is then fed back to the channel estimator for use during subsequent iterations. The turbo decoder can, therefore, be made to work in two modes. First, in a sequential mode, where the number of channel estimator passes, ν, equals one. Second, in a decision-directed mode, where $\nu > 1$. Different combination of η and ν can be specified per application and desired performance.

5.4 Simulation Results

Several computer simulations were performed to study the sensitivity of the sequential and the decision-directed modes of turbo decoding to the granularity of the HMM and the fading rate of the channel. Since each mode can employ either the Viterbi or the forward-backward estimator, we shall refer to the different combinations as the sequential-Viterbi (SV) decoder, the decision-directed-Viterbi (DDV) decoder, and likewise the sequential-forward-backward (SFB) decoder and the decision-directed-forward-backward (DDFB) decoder. All simulations use a random binary PSK modulated transmitted sequence, a rate $r = 1/3$ constraint length $K_c = 3$ turbo code, a 1024 bit pseudo-random interleaver, 8 iterations of Log-MAP iterative decoding, and a stopping criterion of 500 bit errors

5.4.1 Sensitivity to Model Granularity

We first consider the performance of the four aforementioned decoders over a 32-state HMM with a normalized fading rate $f_d T_s = 0.01$. Figure (5.3) shows the bit error rate (BER) vs E_b/N_o plots for the different decoders. Also shown is the ideal BER performance that corresponds to perfect channel side information (SI) cognizance at the receiver. It is evident that beyond an E_b/N_o of 2 dB, the four decoders perform on average 1 dB worse than the ideal case. The figure shows that the SFB decoder slightly out performs its Viterbi counterpart. Interestingly, the 2-pass DDFB decoder performs better than the 2-pass DDV decoder. In fact, the latter performs worse than the SV decoder. Moreover, the figure shows that the 2-pass DDFB decoder and the 4-pass DDV decoder perform comparable to, but no better than their sequential counterparts. The poor performance of the SV and the DDV decoders can be explained by analyzing the performance of the channel estimator. Figure (5.4) shows the estimated noise variance versus E_b/N_o plots for the SV and the DDV decoders. It is evident that the DDV decoders over estimate the noise variance, while the SV decoder under estimates the noise variance. However, at low signal-to-noise ratios, the extent of over estimation is greater than the extent of under estimation, although the

amount of over estimation decreases with increasing number of passes through the channel estimator. Figure (5.5) shows the state estimation error rate (SER) versus E_b/N_o for the SV and the DDV decoders. As expected—based on the results presented in Section (4.4.1)—the Viterbi estimator performs poorly when the fading rate is high (recall that the performance of the forward-backward estimator is similar to that of the Viterbi estimator). The poor BER performance of the decoders can, therefore, be attributed to the combined effect of imperfect noise variance and fading amplitude estimation.

We now consider the performance of the decoders over a 64-state HMM with a normalized fading rate $f_dT_s = 0.01$. Figure (5.6) shows the BER performance of the SV and the DDV decoders. Also shown is the ideal BER performance that corresponds to perfect channel estimation at the receiver. Note that this ideal BER performance seems worse than the ideal performance shown in Figure (5.3). This discrepancy arises from the fact that the higher resolution of the 64-state HMM tantamounts to a large number of states with small fading amplitudes that contribute to deep fades in the channel. Once again, the figure shows that the SV decoder performs better than the 2-pass DDV decoder, and that the 4-pass DDV decoder performs comparable to, but no better than its SV counterpart.

5.4.2 Effect of Fading Rate

We consider the effect of the channel fading rate by analyzing the performance of the decoders over a 32-state HMM with a normalized fading rate $f_dT_s = 0.001$. Figure (5.7) shows the BER performance of the SV, SFB, and the DDV decoders. Both the 2-pass and the 4-pass DDV decoders perform better than the SV decoder. This improved performance can be explained by analyzing the performance of the channel estimator. Figure (5.8) shows the estimated noise variance vs E_b/N_o plots for the three decoders. A comparison of Figures (5.8) and (5.4) show that the extent of noise variance over estimation by the DDV decoders decreases as the fading rate decreases. Moreover at high signal-to-noise ratios, the estimate from the DDV decoders converges to the ideal, while the estimate from the SV decoder diverges from the ideal. Figure (5.9) shows the SER performance of the three decoders. All three decoders exhibit improvements in SER, when compared to decoders used over a channel with a normalized fading rate $f_dT_s = 0.01$. The figure shows that the DDV decoders out perform the SV decoder. However, no appreciable improvement in SER is realized by increasing the number of channel estimator passes from two to four. The improved BER performance of these decoders over a fading channel with a normalized fading rate $f_dT_s = 0.001$ can, therefore, be attributed to improved noise variance and fading amplitude estimation. Figure (5.7) also shows that the SFB decoder out performs it Viterbi counterpart. Since both the Viterbi and the forward-backward channel estimators were shown to perform equally well in Chapter (4), the improved BER performance of the SFB decoder can be attributed to the use of weighted likelihood ratios.

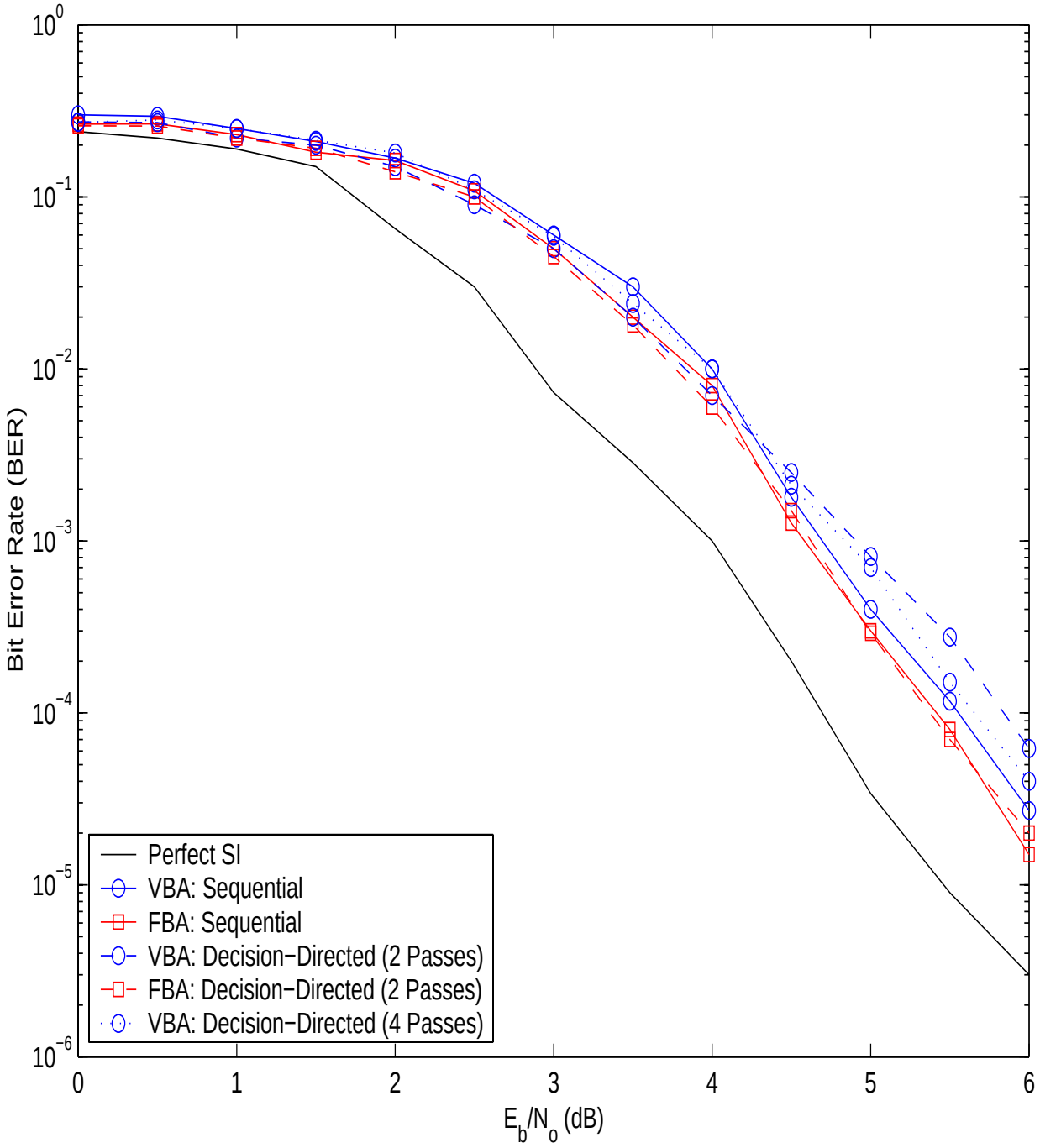

Figure 5.3: Bit error rate performance of different channel estimation and iterative decoding schemes over a 32-state HMM with a normalized fading rate $f_d T_s = 0.01$. The turbo code used is rate $r = 1/3$ with constraint length $K_c = 3$, and uses a 1024 bit pseudo-random interleaver with 8 iterations of Log-MAP decoding.

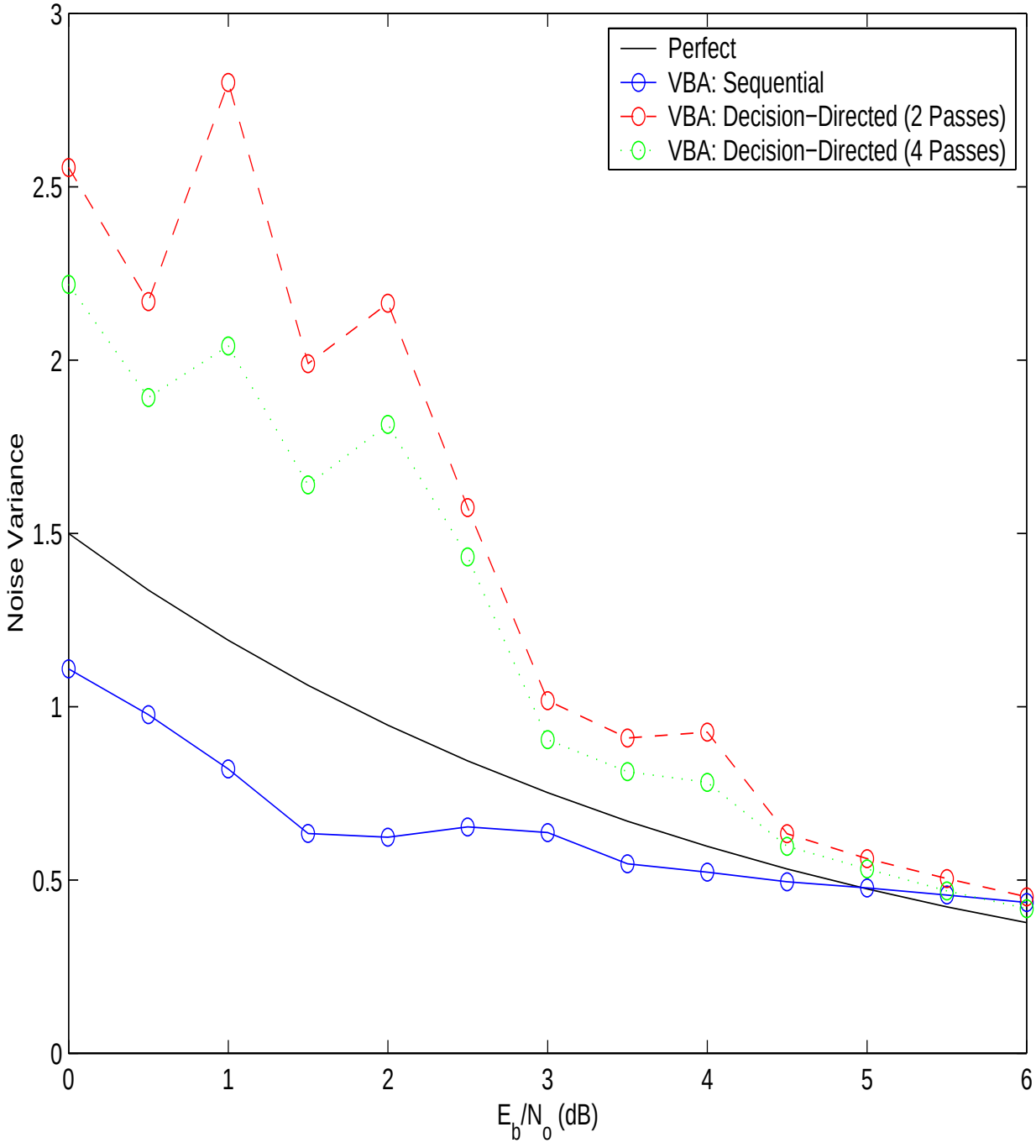

Figure 5.4: Estimated noise variance for the different channel estimation and iterative decoding schemes operating over a 32-state HMM with a normalized fading rate $f_d T_s = 0.01$.

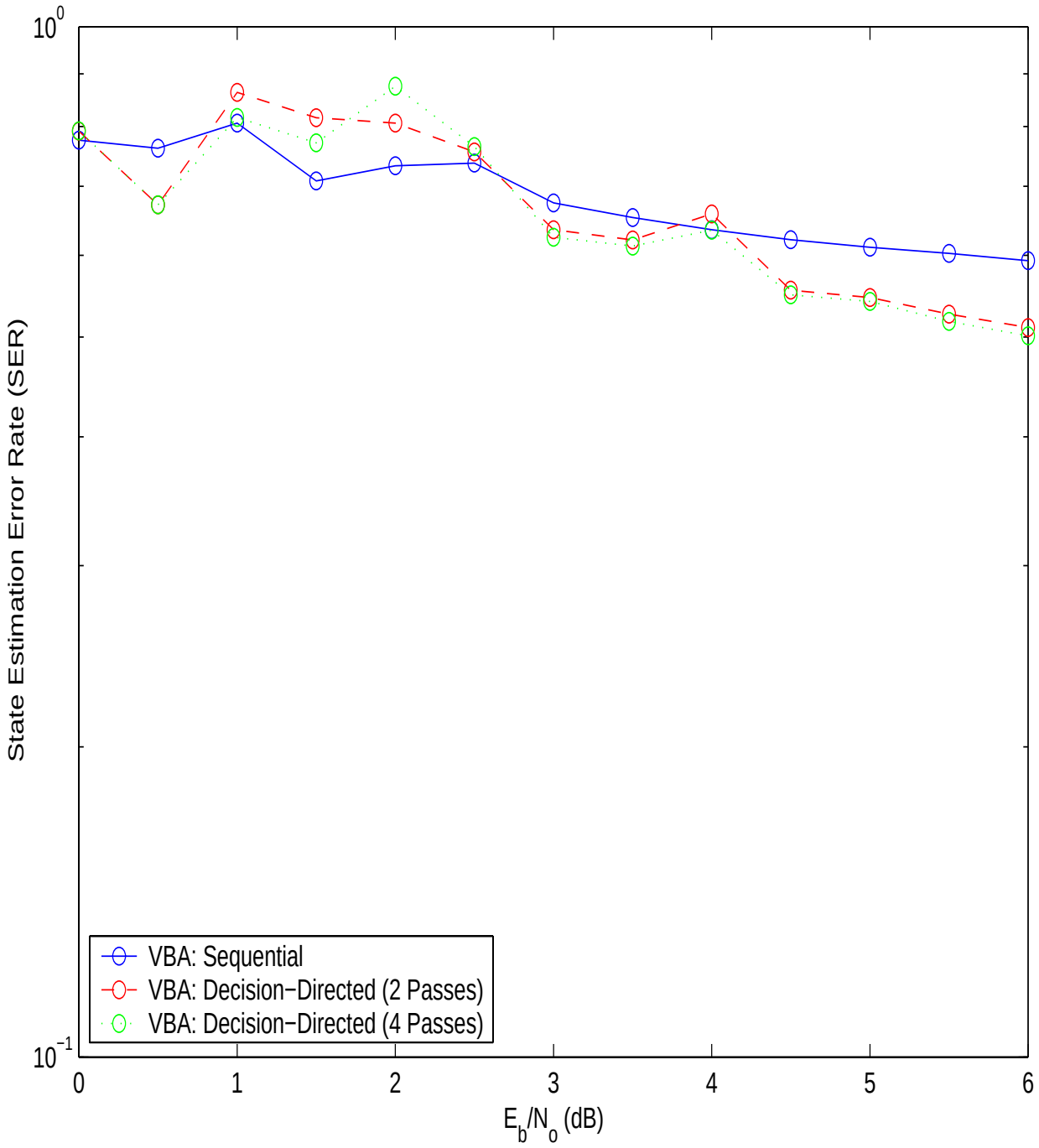

Figure 5.5: Channel state estimation error rate performance of the different channel estimation and iterative decoding schemes operating over a 32-state HMM with a normalized fading rate $f_d T_s = 0.01$.

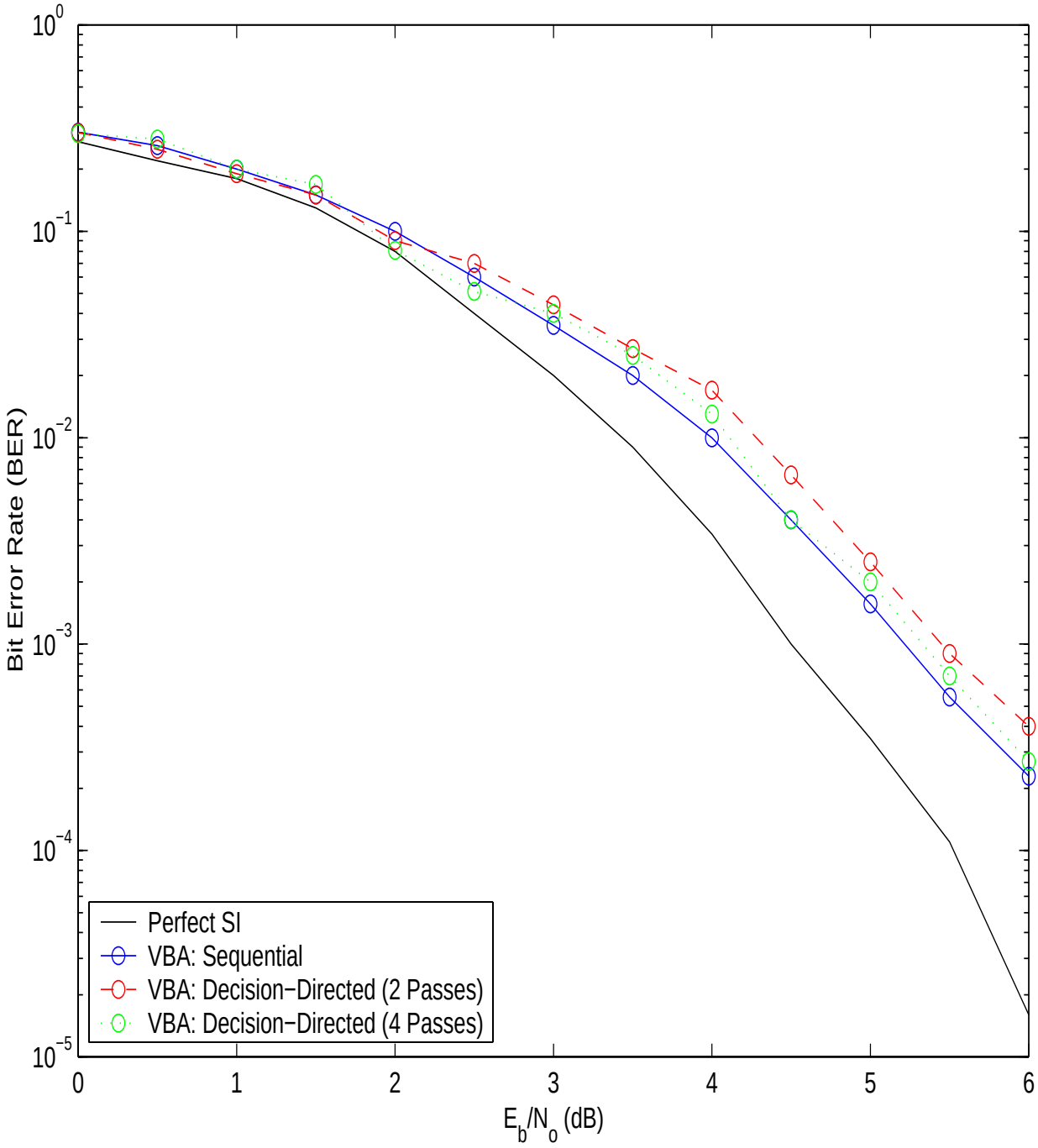

Figure 5.6: Bit error rate performance of different channel estimation and iterative decoding schemes over a 64-state HMM with a normalized fading rate $f_d T_s = 0.01$. The turbo code used is rate $r = 1/3$ with constraint length $K_c = 3$, and uses a 1024 bit pseudo-random interleaver with 8 iterations of Log-MAP decoding.

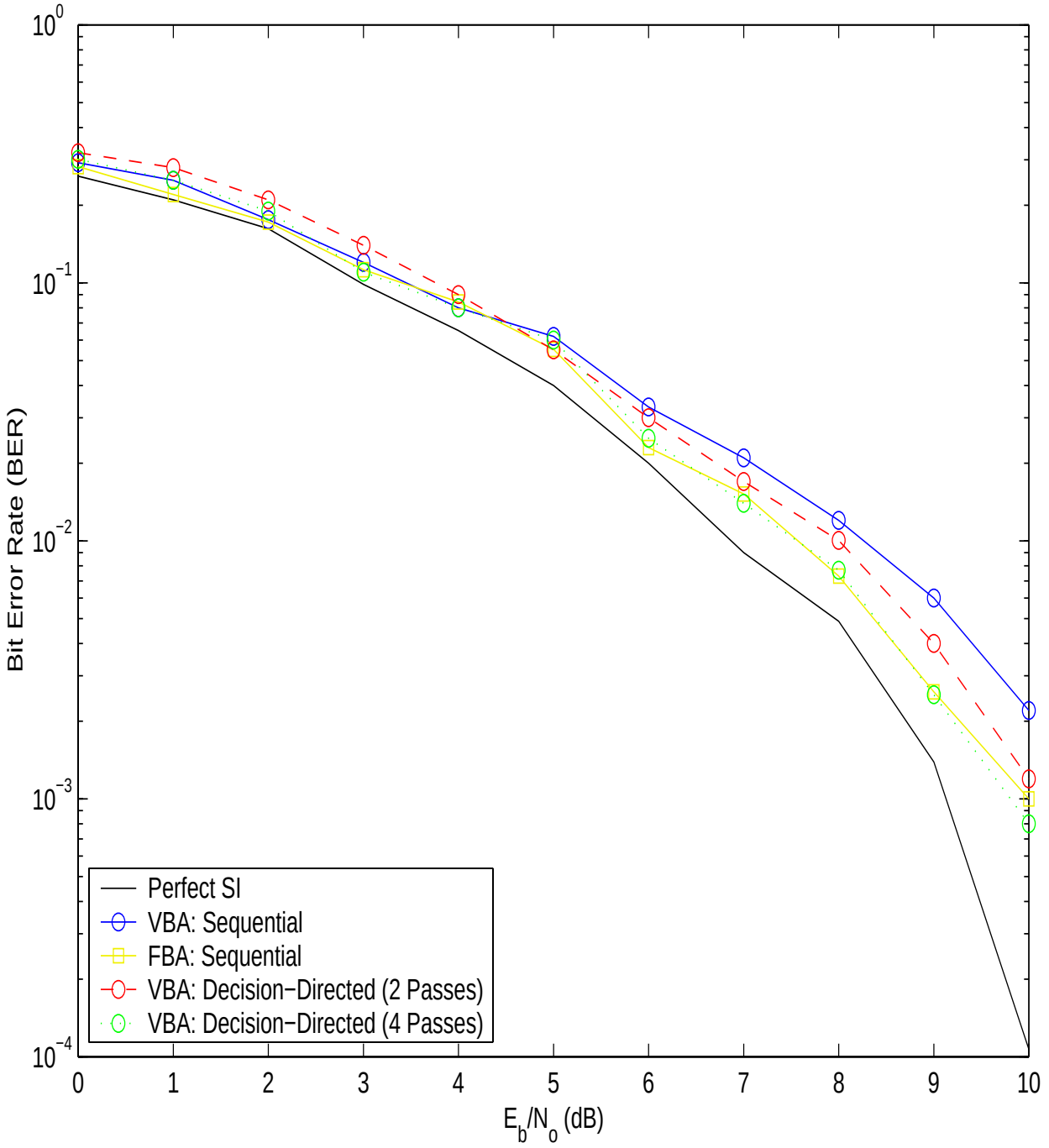

Figure 5.7: Bit error rate performance of different channel estimation and iterative decoding schemes over a 32-state HMM with a normalized fading rate $f_d T_s = 0.001$. The turbo code used is rate $r = 1/3$ with constraint length $K_c = 3$, and uses a 1024 bit pseudo-random interleaver with 8 iterations of Log-MAP decoding.

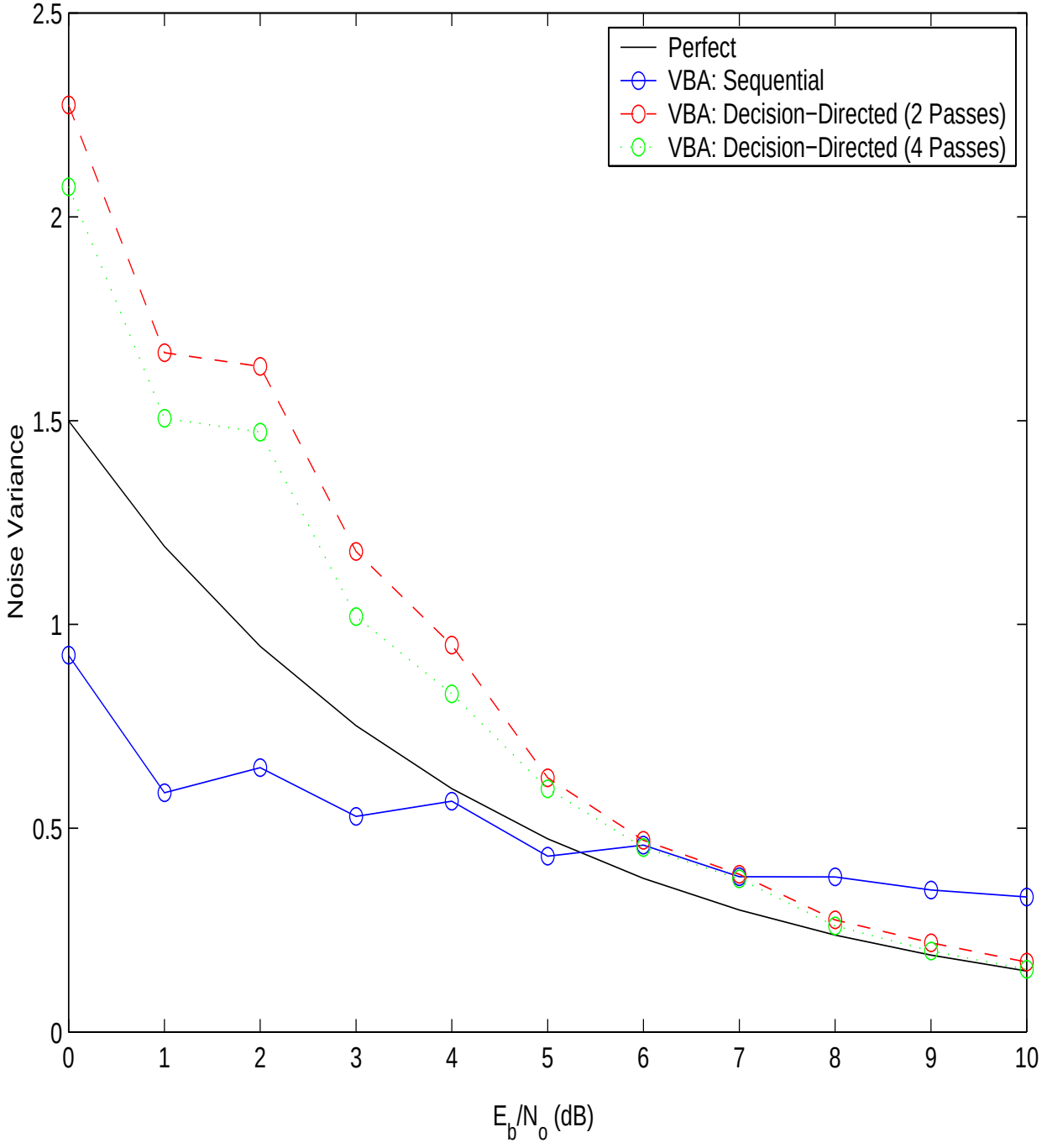

Figure 5.8: Estimated noise variance for the different channel estimation and iterative decoding schemes operating over a 32-state HMM with a normalized fading rate $f_d T_s = 0.001$.

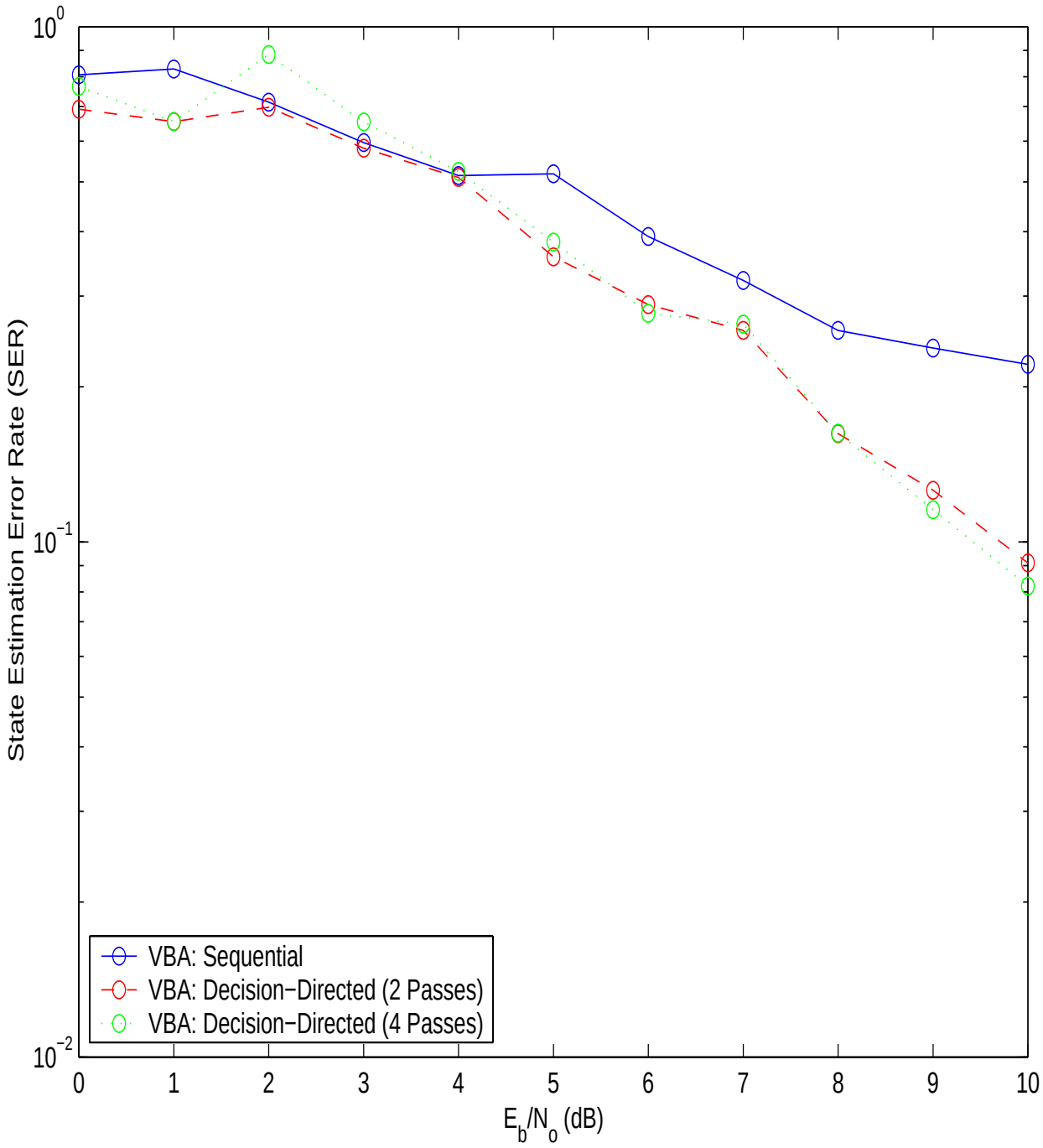

Figure 5.9: Channel state estimation error rate performance of the different channel estimation and iterative decoding schemes operating over a 32-state HMM with a normalized fading rate $f_d T_s = 0.001$.

5.5 Chapter Summary

The computation of the channel reliability metric in iterative decoding schemes necessitate the need for channel estimation. The use of Markov fading channels allows us to formulate a unified trellis based framework, where iterative decoding and channel estimation can either be performed jointly over a hyper-trellis, or separately over their respective trellises. This idea of a unified trellis can be further expanded to include other operations such equalization, multiuser detection, demodulation, etc., that can be described via trellises.

The aim of this chapter was to consolidate the work presented earlier in this thesis on Markov fading channels and trellis based channel estimation techniques with the iterative decoding principle. The chapter presented sequential and decision-directed modes of channel estimation and iterative decoding as sub-optimal but computationally efficient solutions to the problem of joint channel estimation and iterative decoding. It was shown that the sensitivity of the channel estimation algorithms to the granularity of the Markov model and to the fading rate of the channel had a significant impact on the performance of the two aforementioned decoding schemes. However, the chapter described a novel approach of using weighted reliability metrics to compensate the poor performance of the channel estimation algorithms. Future studies should include a more in-depth analysis of joint iterative decoding and channel estimation, should analyze the affect of phase variations on channel estimation and turbo decoding, and should benchmark the performance of the turbo decoder over different types of fading channels.

Bibliography

[1] C.E. Shannon, "A mathematical theory of communication," *Bell Systems Technical Journal*, vol. 27, pp. 379-423, 623-656, July-October 1948.

[2] S. Verdu, "Fifty years of Shannon Theory," *IEEE Transactions on Information Theory*, vol. 44, no. 6, pp. 2057-2078, October 1998.

[3] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near Shannon limit error correcting coding and decoding: Turbo codes (1)," *Proceedings, International Conference on Communications*, pp. 1064-1070, May 1993.

[4] J. Hagenauer, "Source controlled channel decoding," *IEEE Transactions on Communications*, vol. 43, pp. 2449-2457, September 1995.

[5] A. Ruscitto, and E.M. Biglieri, "Joint source and channel coding using turbo codes over rings," *IEEE Transactions on Communications*, vol. 46, no. 8, pp. 981-984, August 1998.

[6] J. Lodge, and M. Gertsman, "Joint detection and decoding by turbo processing for fading channel communications," *Proceedings, International Symposium on Turbo Codes and Related Topics*, pp. 88-95, September 1997.

[7] C. Douillard, M. Jezequel, C. Berrou, A. Picart, P. Didier, and A. Glavieux, "Iterative correction of intersymbol interference: Turbo equalization," *European Transactions on Telecommunications*, vol. 6, pp. 507-511, September/October 1995.

[8] D. Raphaeli, and Y. Zarai, "Combined turbo equalization and turbo decoding," *IEEE Communications Letters*, vol. 2, no. 4, pp. 107-109, April 1998.

[9] D.J. van Wyk, and L.P. Linde, "Turbo coded/multi-antenna diversity combining scheme for DS/CDMA systems," *Proceedings, IEEE International Symposium on Spread Spectrum Techniques and Applications*, vol. 1, pp. 18-22, 1998.

[10] M.C. Valenti, and B.D. Woerner, "Iterative multiuser detection for convolutionally coded asynchronous DS-CDMA," *Proceedings, International Symposium on Personal, Indoor and Mobile Radio Communications*, vol. 1, pp. 213-217, 1998.

[11] P. Hoeher, "On channel coding and multiuser detection for DS-CDMA," *Proceedings, International Conference on Universal Personal Communications*, pp. 641-646, October 1993.

[12] X. Wang, and H.V. Poor, "Iterative (turbo) soft interference cancellation and decoding for coded CDMA," *IEEE Transactions on Communications*, vol. 47, no. 7, pp. 1046-1060, July 1999.

[13] P.D. Alexander, M.C. Reed, J.A. Asenstorfer, and C.B. Schlegel, "Iterative multiuser interference reduction: Turbo CDMA," *IEEE Transactions on Communications*, vol. 47, no. 7, 1008-1013, July 1999.

[14] V. Kuhn, "Turbo codes and turbo-coded modulation in CDMA mobile radio systems with short frame transmission," *Proceedings, IEEE International Symposium on Spread Spectrum Techniques and Applications*, vol. 2, pp. 364-386, 1998.

[15] T.M. Duman, and M. Salehi, "Performance bounds for turbo-coded modulation systems," *IEEE Transactions on Communications*, vol. 47, no. 4, pp. 511-521, April 1999.

[16] A.H.S. Mohammadi, and W. Zhuang, "Combined turbo code and modulation for CDMA wireless communications," *Proceedings, IEEE Vehicular Technology Conference*, vol. 3, pp. 1920-1924, 1998.

[17] P. Hoeher, and J. Lodge, "Turbo DPSK: Iterative differential PSK demodulation and channel decoding," *IEEE Transactions on Communications*, vol. 47, no. 6, pp. 837-842, June 1999.

[18] M. A. Jordan, and R. A. Nichols, "The effects of channel characteristics on turbo code performance," *Proceedings, IEEE MILCOM*, pp. 17-21, 1996.

[19] M. C. Reed, and J. Asenstorfer, "A novel variance estimator for turbo code decoding," *Proceedings, International Conference on Telecommunications*, pp. 173-178, April 1997.

[20] T. A. Summers, and S. G. Wilson, "SNR mismatch and online estimation in turbo decoding," *IEEE Transactions on Communications*, vol. 46, pp. 421-423, April 1998.

[21] J. K., Cavers, "An analysis of pilot symbol assisted modulation for Rayleigh fading channels," *IEEE Transactions on Vehicular Technology*, vol. 40, pp. 686-693, November 1991.

[22] J. K. Cavers, and M. Liao, "A comparison of pilot tone and pilot symbol techniques for digital mobile communications," *Proceedings, IEEE Global Telecommunications Conference*, pp. 915-921, December 1992.

[23] M. C. Reed, and J. Asenstorfer, "A novel variance estimator for turbo code decoding," *Proceedings, International Conference on Telecommunications*, pp. 173-178, April 1997.

[24] C. Schlegel, and L. Perez, "On error bounds and turbo codes," *IEEE Communications Letters*, vol. 3, no. 7, pp. 205-207, July 1999.

[25] C. Berrou, and A. Glavieux, "Near optimum error correcting codes and decoding: turbo-codes," *IEEE Transactions on Communications*, vol. 44, no. 10, pp. 1261-1271, October 1996.

[26] D. Divsalar, and E. Pollara, "Low rate turbo codes for deep space communications," *Proceedings, IEEE International Symposium on Information Theory*, pp. 35, 1995.

[27] J.D. Andersen, "Turbo coding for deep space applications," *Proceedings, IEEE International Symposium on Information Theory*, pp. 36, 1995.

[28] D. Divsalar, and F. Pollara, "Multiple turbo codes for deep space communications," *JPL TDA Progress Report*, vol. 42, May 1995, pp. 66-77.

[29] J.C. Morakis, and W.H. Miller, "Coding techniques under study at NASA," *Preceedings, IEEE Aerospace Conference,*, pp. 559-565, 1997.

[30] M. Penicaud, Y. Yongacoglu, and J.Y. Chauinard, "Iterative decoding of rate adaptve codes for mobile satellite fading channels," *Proceedings, International Symposium on Turbo Codes and Related Topics*, pp. 231-234, September 1997.

[31] P. Robertson, and T. Worz, "Bandwidth efficient turbo trellis-coded modulation using punctured component codes, "*IEEE Journal on Selected Areas of Communications*, vol. 16, pp. 206-218, February 1998.

[32] B. Melis, and G. Romano, "Application of turbo codes in the uplink of a DS-CDMA system," *Proceedings, International Symposium on Turbo Codes and Related Topics*, pp. 243-246, September 1997.

[33] L. Bomer, F. Burket, J. Eichinger, R. Halfmann, W. Liegel, and M. Werner, "A CDMA radio link with turbo decoding: concept and performance evaluation," *Proceedings, IEEE Personal Indoor and Mobile Radio Communications Conference*, pp. 788-793, September 1995.

[34] P. Jung, J. Plechinger, M. Doetsch, and F.M. Berens, "Advances on the application of turbo codes to data services in third generation mobile networks," *Proceedings, International Symposium on Turbo Codes and Related Topics*, pp. 135-142, September 1997.

[35] M.C. Valenti, and B.D. Woerner, "Variable latency turbo codes for wireless multimedia applications," *Proceedings, International Symposium on Turbo Codes and Related Topics*, pp. 216-219, September 1997.

[36] M. Zeng, A.Annamalai, and V.K. Bhargava, "Recent advances in cellular wireless communications," *IEEE Communications Magazine*, vol. 39, no. 9, pp. 128-138, September 1999.

[37] S. Benedetto, and G. Montorsi, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *JPL TDA Progress Report*, vol. 42, August 1996.

[38] S. Benedetto, and G. Montorsi, "Generalized concatenated codes with interleavers," *Proceedings, International Symposium on Turbo Codes and Related Topics*, pp. 32-39, September 1997.

[39] O.Y. Takeshita, O. Collins, P.C. Massey, and D.J. Costello, "A note of asymmetric turbo codes," *IEEE Communications Letters*, vol. 3, no. 3, pp. 69-71, March 1999.

[40] M.C. Valenti, *Iterative detection and decoding for wireless communications*, PhD thesis, Virginia Polytechnic Institute and State University, July 1999.

[41] D. Divsalar, and F. Pollara, "On the design of turbo codes," *JPL TDA Progress Report*, vol. 42, pp. 99-120, November 1995.

[42] E. Dunscombe, and F.C. Piper, "Optimal interleaving scheme for convolutional codes," *Electronics Letters*, vol. 25, no. 22, pp. 1517-1518, October 1989.

[43] J. Yuan, B. Vucetic, and W. Feng, "Combined turbo codes and interleaver design," *IEEE Transactions on Communications*, vol. 47, no. 49, pp. 484-487, April 199.

[44] D. Divsalar, and F. Pollara, "Turbo codes for PCS applications," *Proceedings, International Conference on Communications*, 1995.

[45] A.S. Barbulescu, and S.S. Pietrobon, "Interleaver design for turbo codes," *Electronics Letters*, vol. 30, pp. 2107-2108, December 1994.

[46] J.C. Cress, and W.J. Ebel, "Turbo code implementation issues for low latency, low power applications," *Proceedings, Virginia Tech Symposium Wireless Personal communications*, pp. 191-200, June 1998.

[47] S. Benedetto, and G. Montorsi, "Design guidelines of parallel concatenated convolutional codes," *Proceedings, IEEE Global Telecommunications Conference*, vol. 3, pp. 2273-2277, November 1995.

[48] S. Benedetto, and G. Montorsi, "Role of recursive convolutional codes in turbo codes," *Electronics Letters*, vol. 31, no. 11, pp. 858-859, May 1995.

[49] S. Benedetto, and G. Montorsi, "Design of parallel concatenated convolutional codes," *IEEE Transactions on Communications*, vol. 44, no. 5, 591-600, May 1996.

[50] J. Hagenauer, "Iterative decoding of binary block and convolutional codes," *IEEE Transactions on Information Theory*, vol. 42, pp. 429-445, March 1996.

[51] J. Hagenauer, and P. Hoeher, "A Viterbi algorithm with soft decision outputs and its application," *Proceedings, IEEE Global Telecommunications Conference*, pp. 1680-1686, 1989.

[52] L. Papke, P. Robertson, and E. Villebrum, "Improved decoding with the SOVA in paralled concatenated (turbo code) scheme," *Proceedings, International Conference on Communications*, pp. 102-106, 1996.

[53] L.R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Transactions on Information Tneory*, pp. 284-287, March 1974.

[54] P. Robertson, P. Hoeher, and E. Villebrum, "Optimal and sub-optimal maximum a posteriori algorithms suitable for turbo decoding," *European Transactions on Communications*, vol. 8, pp. 119-125, March/April 1997.

[55] A. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE Journal on Selected Areas of Communications*, pp. 260-264, Feburary 1998.

[56] A. Viterbi, "Error bounds for convolutional codes and an asymtotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 260-269, April 1967.

[57] E.N. Gilbert, "Capacity of a burst-noise channel," *Bell systems Technical Journal*, vol. 39, pp. 1253-1265, September 1960.

[58] B.D. Fritchman, "A binary channel characterization using partitioned Markov chains," *IEEE Transactions on Information Theory*, vol. IT-13, pp. 221-227, April 1967

[59] W. Turin, and M.M. Sondhi, "Modeling error sources in digital channels," *IEEE Journal on Selected Areas in Communications*, vol. 11, no. 3, pp. 340-347, April 1993.

[60] W. Turin, *Digital Transmission Systems: Performance Analysis and Simulation*, New York: McGraw-Hill, 1998.

[61] I.B. Collings, and J.B. Moore, "An HMM approach to adaptive demodulation of QAM signals in fading channels," *International Journal on Adaptive Control and Signal Processing*, vol. 8, pp. 457-474, 1994.

[62] J.A.R. Fonollosa, and J. Vidal, "Application of hidden Markov models to blind channel chanracterization and data detection," *Proceedings, IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 185-188, April 1994.

[63] C. Anton-Haro, J.A.R. Fonollosa, and J.R. Fonollosa, "Blind channel estimation and data detection using hidden Markov," *IEEE Transactions on Signal Processing*, vol. 45, no. 1, pp. 241-246, January 1997.

[64] R.L. Streit, and R. Barrett, "Frequency line tracking using hidden Markov models," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 38, pp. 586-598, 1990.

[65] C.L. Carlemalm, V. Krisnamurthy, and K. Dogancay, "Algorithms for blind detection of equalization errors in hidden Markov model channels," *Proceedings, IEEE International Conference on Communications*, vol. 1, pp. 324-328, 1998.

[66] F. Swarts, and H.C. Ferreira, "Markov characterization of channels with soft decision outputs", *IEEE Transactions on Communications*, vol. 41, no. 5, pp. 678-682, May 1993.

[67] J. Garcia-Frias, and P.M. Crespo, "Hidden Markov models for burst error characterization in indoor radio channels", *IEEE Transactions on Vehicular Technology*, vol. 46, no. 4, pp. 1006-1019, November 1997.

[68] S. Sivaprakasam, and K.S. Shanmugan, "An equivalent Markov model for burst errors in digital channels," *IEEE Transactions on Communications*, vol. 43, no. 2/3/4, pp. 1347-1355, Feburary/March/April 1995.

[69] S. Sivaprakasam, and K.S. Shanmugan, "A forward-only recursion based HMM for modeling burst errors in digital channels," *Proceedings, IEEE Global Telecommunications Conference*, vol. 2, pp. 1054-1058, 1995.

[70] M. Sajadieh, F.R. Kschischang, and A. Leon-Garcia, "A block memory model for correlated Rayleigh fading channels," *Proceedings, IEEE International Conference on Communications*, pp. 282-286, June 1996.

[71] H.S. Wang, and P.C. Chang, "On verifying the first-order Markovian assumption for a Rayleigh fading channel model," *IEEE Transactions on Vehicular Technology*, vol. 45, pp. 353-357, May 1996.

[72] W. Turin, and R. Nobelen, "Hidden Markov modeling of flat fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 9, pp. 1809-1817, December 1998.

[73] R.A. Howard, *Dynamic Probablistic Systems*, New York: John Wiley and Sons, Inc., 1971.

[74] T.S. Rappaport, *Wireless Communications: Principles and Practice*, New Jersey: Prentice Hall, 1996.

[75] J.G. Proakis, *Digital Communications*, New York: McGraw-Hill, 1995.

[76] R.H. Clarke, "A statistical theory of mobile radio reception," *Bell systems Technical Journal*, vol. 47, pp. 957-1000, 1968.

[77] T.K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, pp. 47-60, November 1996.

[78] L.E. Baum, and T. Petrie, "A maximization technique occurring in the statistical analysis of probablistic functions of Markov chains," *The Annals of Mathematical Statistics*, vol. 41, no. 1, pp. 164-171. 1970.

[79] W. Turin, "Unidirectional and parallel Baum-Welch algorithms," *IEEE Transactions on Speech and Audio Processing*, vol. 6, no. 6, pp. 516-523, November 1998.

[80] L. Wilhelmsson, and L.B. Milstein, "On the Effect of Imperfect Interleaving for the Gilber-Elliot channel," *IEEE Transactions on Communications*, vol. 478, no. 5, pp. 681-688, May 1999.

[81] H. Kong, and E. Shwedyk, "Sequence estimation for frequency non-selective channels using a hidden Markov model," *Proceedings, IEEE Vehicular Technology Conference*, pp. 1251-1253, June 1994.

[82] H. Kong, and E. Shwedyk, "A hidden Markov model based MAP receiver for Nakagami fading channels," *IEEE International Symposium on Information Theory*, pp. 210, 1995.

[83] H. Kong, and E. Shwedyk, "Sequence detection and channel state estimation over finite state markov channels," *IEEE Transactions on Vehicular Technology*, vol. 48, no. 3, pp.833-839, May 1999.

[84] P. Jung, "Camparison of turbo code decoders applied to short frame transmission system," *IEEE Journal on Selected Areas in Communications*, vol. 14, pp. 530-537, April 1996.

[85] E. K. Hall, and S. G. Wilson, "Design and analysis of turbo codes on Rayleigh fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 160-174, February 1998.

[86] K. Koorapaty, Y. P. E. Wang, and K. Balachandran, "Performance of turbo codes with short frame sizes," *Proceedings, IEEE Vehicular Technology Conference*, pp. 329-333, 1997.

Vita

Anwer A. Khan was born in Karachi, Pakistan on August 30, 1975. He received the Bachelor of Science degree in Electrical Engineering from Virginia Polytechnic Institute and State University in May 1998. During the summer of 1998, he became a member of the Center for Wireless Telecommunications (CWT). At CWT he developed a simulation test-bed for turbo codes. In Fall 1999, he became a member of the Mobile and Portable Radio Research Group (MPRG), and continued his research on turbo codes, channel estimation, and hidden Markov models. His research interest include digital signal processing, channel coding, and simulation of digital communication systems.

Anwer is a student member of the IEEE, and a member of the Eta Kappa Nu Electrical Engineering honor society. He has served as a delegate to the Graduate Student Assembly, and has served as a student representative on the Bradley Department of Electrical and Computer Engineering's graduate administration committee, and on the College of Engineering's graduate student committee.