Markov Modeling of Third Generation Wireless Channels

Ihsan A. Akbar

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

Master of Science
in
Electrical Engineering

William H. Tranter, Chair
Brian D. Woerner
Jeffrey H. Reed

March 15, 2003
Blacksburg, Virginia

Keywords: Hidden Markov Models, Semi-Hidden Markov Models, Fading Channels, Error Source Modeling

Markov Modeling of Third Generation Wireless Channels

Ihsan A. Akbar

(ABSTRACT)

Wireless has proved to be one of the most important and fastest growing fields of communications especially during last few decades. To achieve reliable communication, we model a wireless system to analyze its performance and to find ways to improve the reliability of a particular system. Extensive research is being done to accurately model wireless systems, and to achieve better performance. Simulation techniques have been in use for many years to support the design and evaluation of electronic communication systems. Over the past few decades, Computer Aided Design (CAD) techniques (including both computerized analytical techniques and simulation) have matured, and are now usually applied at some point in the system design/development process.

The aim of this thesis is to find efficient algorithms that can model third generation wireless channels in a discrete sense. For modeling these channels, mathematical tools known as hidden Markov models are used. These models have proved themselves to be very efficient in many areas of electrical engineering including speech recognition, pattern recognition, artificial intelligence, wavelets and queuing theory. Wideband Code Division Multiple Access (W-CDMA) wireless communication parameters including channels fading statistics, Bit Error Rate (BER) performance and interval distribution of errors are modeled using different Markov models, and their results are tested and validated.

Four algorithms for modeling error sources are implemented, and their results are discussed. Both hidden Markov models and semi-hidden Markov models are used in this thesis, and their results are validated for the W-CDMA environment. The state duration distributions for these channels are also approximated using Phase-Type (PH) distribution.

Acknowledgements

All praises and thanks are to God the one, the only creator and the ruler of this universe. I wish I could thank Him for what I am and for all He has granted me although He can never be thanked enough.

Many thanks and gratitude to my mentor Dr. William H. Tranter, without whom I would not have been able to accomplish my M.S. research work at Virginia Tech. I thank him for advising me during my research, counseling me during my studies, and providing a friendly and supportive environment for me to carry out my M.S. research.

I want to show gratitude to my committee members and other teachers of Virginia Tech. Their dedication and expertise in their respective fields are and will always be a source of inspiration for me.

During our course of life we encounter many people who help us, guide us, share their valuable time with us and provide support to us so that we can achieve our goals of life. I would like to thank all of them for being so helpful to me.

I am thankful to God who has blessed me with so many nice friends. My friends are there for me whenever I need them. I thank them for their encouragement, kind support and cooperation. Many thanks to my co-worker and the staff at the Mobile and Portable Radio Research Group (MPRG) of Virginia Tech for sharing their research work and helping me to accomplish my M.S. research.

I would like to mention the continued support and love of my wife, Fatima Zuberi, and would like to thank her for being so understanding with me. It is her continued caring and encouragement that keep me motivated with what I do and I pray to God that we will always live a very pleasant and joyous life.

I cannot find the words to express my profound feelings of gratitude for my parents, my father Inamuddin Akbar, my mother Tahira Darakhshan, and other family members. I need their constant love, help, support and guidance before taking any major step of my life.

Contents

List of Figures

List of Tables

Chapter

1

Introduction

The problem of channel modeling is one of the most challenging and computationally inefficient steps in modeling a digital communication system. A typical digital communication system consists of a data source and input transducer, a source encoder, a channel encoder, a digital modulator, a channel, a digital demodulator, a channel decoder, a source decoder and output transducer. A reliable computer simulation of all these parts of a communication system is required to accurately analyze the performance of digital communication systems. Waveform level simulation is generally used for simulating different parts of a digital communication system.

In this thesis a different approach will be applied to model digital communication channels. This is especially applicable to the third generation wireless (mobile) channels that usually contain many interferers and multipath. We will use discrete channel modeling techniques to model these complex channels, and hidden Markov models [1] will be used as a mathematical tool for modeling and simulation. These models were introduced by Shannon [3] as finite state channels. They are also known as *discrete channel models*, *stochastic sequential machines* and *probabilistic automata*. Discrete channel models can be memoryless or may contain memory. Memoryless channels do not produce inter-symbol interference (ISI) or fading. Modeling discrete memoryless channels is a simple and straightforward process, whereas modeling a discrete channel with memory is relatively complex. Wireless communication channels generally have

memory and correlation between input and output symbols. Fading wireless channels are good examples of channels having correlated errors.

Figure 1.1.1 shows a simplified diagram of a digital communication system. A waveform channel model is shown that generally contains distortion, fading, noise and interference. These disturbances combine with the transmitted signal to define the waveform at the input to the receiver.

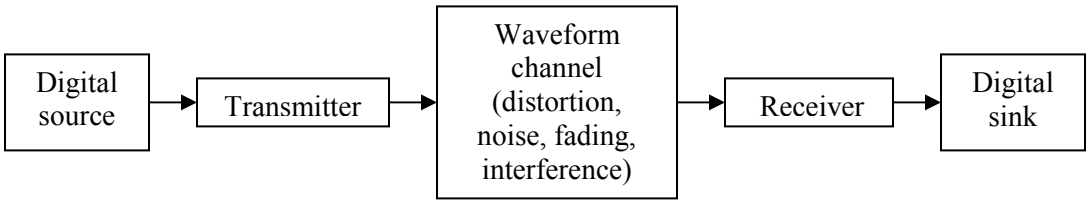

Figure 1.1.1: Block diagram of a digital communication system

In this thesis we will investigate the methods that can replace waveform channel models with discrete channel models. These discrete channel models are computationally more efficient and take less runtime than the conventional waveform channel models. Once a reliable mode is obtained, the waveform channel model can be replaced with its equivalent discrete channel model.

Figure 1.1.2 shows a binary symmetric channel. The input and output alphabets are both binary i.e. 0 or 1, and the channel transition probabilities are symmetrical. It can be stated that $\Pr(0|0) = \Pr(1|1) = (1-p)$ and $\Pr(1|0) = \Pr(0|1) = p$, where p is the error probability.

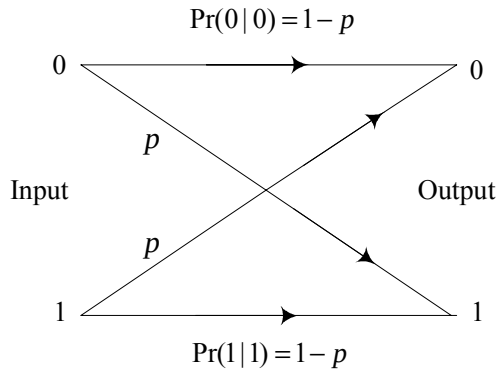

Figure 1.1.2: Binary symmetric channel transition diagram

The channel is symmetric in that its zeroes and ones are affected by the channel in the same manner. Note that for a memoryless channel, the error probability will affect every symbol in the same manner. However for the channel having memory, the error probability depends upon the previous bits transmitted. Figure 1.1.3 shows a discrete m-ary input and M-ary output channel. In this case $\Pr(Y = y_i \mid X = x_j) = \Pr(y_i \mid x_j)$

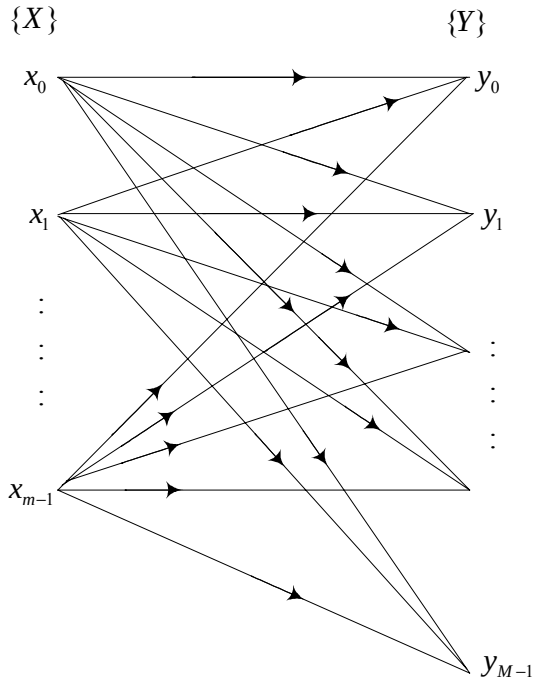

Figure 1.1.3: Discrete m-ary input, M-ary output channel

Figure 1.1.2 and 1.1.3 are examples of discrete channel models that are memoryless. Let us now focus on the more interesting case in which memory effects are contained in channel models.

1.1 Summary

In this chapter a basic idea about discrete channel models and their advantages in modeling and simulations is given. Memoryless channels and channels that contain memory are also briefly discussed in this chapter. Note however that for replacing a waveform level channel with its discrete counterpart, we always need to run the waveform level channel model at least once.

Chapter

2

Basic Theory

This chapter deals with the fundamental concepts related to Markov processes. We start with the notion of state transitions, and then the well-known Markovian assumption is discussed. The transition probabilities, the state transition matrix concept and their representations are discussed later. A brief description about hidden Markov models (HMMs) and the representation of wireless channels as finite state channels are given and some basic models are discussed in later sections. Semi-hidden Markov models (SHMMs), their matrix representation, and their use in wireless channel modeling are studied in detail. Next the issue of error source simulation is addressed. Fading wireless channels are discussed in the last section of this chapter. Some examples are included for better illustration of these fundamental concepts.

2.1 The State and Transition Concept

We can specify a physical system by giving the values of number of variables that describe the system [1]. For example, a chemical system can often be specified by the values of temperature, pressure, and volume, whereas the instantaneous description of a spacecraft would include its position in spatial coordinates, its mass, and its velocity. These critical variables of a system are called *state* variables. When the values of all state variables of a system are known, we can say that its state has been specified. Thus we can say that the state of a system is all that is needed in order to describe the system at any

instant. In the course of time a system passes from state to state and thus exhibits dynamic behavior. Such changes are called *state-transitions* or simply transitions.

Let us start with an example of a frog and a lily pond [1]. Assume that there are a large number of lily pads in a pond, and a frog is sitting on one of these pads. Let us also assume that the frog must always sit on a pad and never swim in the water. The frog occasionally jumps in the air and lands on a lily pad; sometimes the one it had been sitting on and sometimes a different one. For the moment we are interested not in the time pattern of its jumping, but in the location of the frog after successive jumps.

To make our discussion specific, lets consider that the pond has a finite number of lily pads, N, numbered from 1 to N. If we use n to denote the number of jumps made by the frog, and s_n to denote the number of the pad occupied after the nth jump, then the sequence

$$s_0, s_1, s_2, \ldots, s_n, s_{n+1}$$

specifies the frog's itinerary. Such a sequence is known as the trajectory of the process.

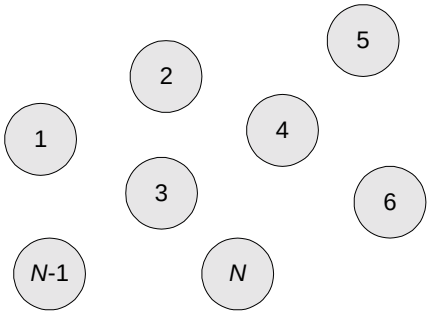

Figure 2.1.1: The pond abstracted

We now have a physical realization of a *state-transition* process. The state of the process is the number of the pad occupied by the frog. The quantity s_n represents the state of the process after its nth transition, which is called its state at time index n. In an N-state process, s_n may take on any integer value $1, 2, \ldots, N$ at any time index $n = 0, 1, 2, \ldots$ that

shows the number of transitions made by the process. Figure 2.1.1 shows the abstraction of the pond.

As we observe the sequence of the states occupied by the frog, i.e., the process trajectory, it can be easily observed that it is random. The statistical behavior of this process could be specified if the probability

$$\Pr(s_{n+1} = j \mid s_n = i, s_{n-1} = k, \ldots, s_0 = m) \tag{2.1.1}$$

was known for all values of its arguments. This is the probability that each state will be occupied after the $n+1$th transition, given the entire trajectory or history of state occupancies through time n.

2.2 The Markovian Assumption

The Markovian assumption can be described as: *Only the last state occupied by the process is relevant in determining its future behavior*. Hence all the previous states before the present state are not significant in determining the future state of the system. This greatly simplifies both the possible behavior of the process and the problem of specifying the process. Mathematically

$$\Pr(s_{n+1} = j \mid s_n = i, s_{n-1} = k, \ldots, s_0 = m) \tag{2.2.1}$$

becomes

$$\Pr(s_{n+1} = j \mid s_n = i) \tag{2.2.2}$$

Thus, the probability of making a transition to each state of the process depends only on the state presently occupied. Equivalently, the future trajectory of a process depends only on its present state.

This is a very strong assumption only few physical systems can expect to satisfy this assumption in a strict sense. Yet the Markov process is an extremely useful model for wide classes of systems. We will try to apply this assumption to modeling wireless channels and its validity in different cases.

Note that no experiment can ever show the ultimate validity of the Markovian assumption [7]; hence, no physical system can ever be classified absolutely as either Markovian or non-Markovian. Now the important question is whether the Markov model is useful, in that it provides accurate results. Then the Markov assumption can be justified, and the investigator can enjoy analytical and computational convenience not often found in complex models.

Consider a system where the states before the last state occupied also influence future behavior. We can still specify the Markovian assumption for these systems by changing the state structure. Suppose that the last two states occupied both influence the transition to the next state. Then a new process can be defined with N^2 states - each state in the new process would correspond to a pair of successive states in the old process. With this redefinition, the property of (2.2.2) could be satisfied, but at the expense of a considerable increase in computational complexity [6]. In literature, this is also known as the *curse of dimensionality*. Any dependence of future behavior on a finite amount of past history can, at least theoretically, be treated in the same way. Many references are available that discuss the Markovian assumption in considerable detail and some of them are [1, 4, 5, 6, 10, 11, 22, 36, 52, 55, 60 and 68]. To study the flow graph methods of Markovian assumptions, refer to [6] and [69].

2.3 Transition Probabilities

The basic requirement for defining a Markov process is to specify the probability of making the next transition state for each state in the process and for each transition time. Thus the quantity

$$\Pr(s_{n+1} = j \mid s_n = i) \tag{2.3.1}$$

must be specified as $1 \le i,\ j \le N$, and for $n = 0, 1, 2, \ldots$ where $i,\ j$ denote the state index, and n denotes the discrete time index. Our notation allows the possibility that each transition to change as successive transitions are made. We thus define the transition probability p_{ij} as

$$p_{ij} = \Pr(s_{n+1} = j \mid s_n = i) \qquad 1 \le i,\ j \le N,\ \ n = 0, 1, 2, \ldots \quad (2.3.2)$$

The transition probability p_{ij} is the probability that a process presently in state i will occupy state j after its next transition. Since the transition probability p_{ij} is a probability, it must satisfy the requirement,

$$0 \le p_{ij} \le 1 \qquad 1 \le i,\ j \le N \quad (2.3.3)$$

and that process must occupy one of its N states after each transition,

$$\sum_{j=1}^{N} p_{ij} = 1 \qquad i = 1, 2, \ldots, N \quad (2.3.4)$$

The N^2 transition probabilities that describe a Markov process are conveniently represented by a $N \times N$ transition probability matrix, generally denoted by $\mathbf{P}$ in the literature, having elements

$$\mathbf{P} = (p_{ij}) = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1N} \\ p_{21} & p_{22} & \cdots & p_{2N} \\ : & & & : \\ : & & & : \\ p_{N1} & p_{N2} & \cdots & p_{NN} \end{bmatrix} \quad (2.3.5)$$

The entries in $\mathbf{P}$ must satisfy the requirements imposed by (2.3.3) and (2.3.4). The matrices whose elements cannot lie outside the range $(0,1)$ and whose rows sum to one, fall in the special category of matrices known as *stochastic matrices* [18]. Since the rows of the transition probability matrix sum to one, only $N(N-1)$ parameters are necessary to specify the transition probabilities behavior of a N-state Markov process.

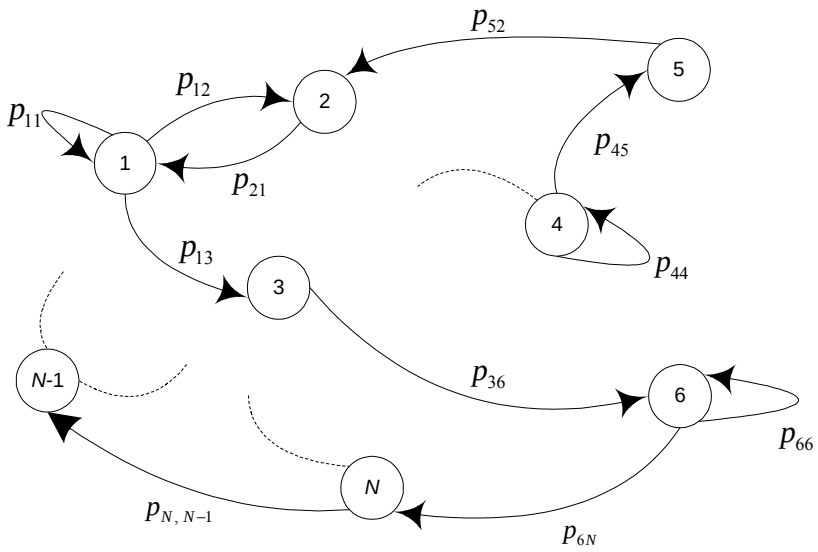

Figure 2.3.1: Different states and their transitions

The transition probability matrix of a Markov process, and hence the process itself, can be graphically represented by a transition diagram like that shown in Figure 2.3.1. Every node is numbered to represent one state of the process. A directed line segment, or branch, is drawn from each node i to each node j, with the transition probability p_{ij}. In numerical examples we shall use the convention that only those directed line segments corresponding to non-zero transition probabilities will be shown in the transition diagram.

Example 2.3.1

This example shows how to obtain the probability of a particular Markov chain. Suppose there is a three state Markov process in which state 1 denotes *Rainy* (R), state 2 denotes *Cloudy* (C) and state 3 denotes *Sunny* (S) weather. The state transition probability matrix is given as

$$\mathbf{P} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

Compute the probability of observing SSRSCS given that today is S.

Using the chain rule, we obtain

$$\Pr(S,S,R,S,C,S) = \Pr(S)\Pr(S|S)\Pr(R|S)\Pr(S|R)\Pr(C|S)\Pr(S|C)$$

$$= \pi_3 a_{33} a_{31} a_{13} a_{32} a_{23}$$

$$= 1(0.8)^2(0.1)(0.3)(0.1)(0.2) = 0.000384$$

2.4 Hidden Markov Models (HMMs)

Let us look at a simple example to understand the notion of hidden Markov models (HMMs). This example is inspired from [22]. Suppose that there is an urn that contains balls having three different colors distributed in different proportions. Suppose also that these colors are white, gray and black.

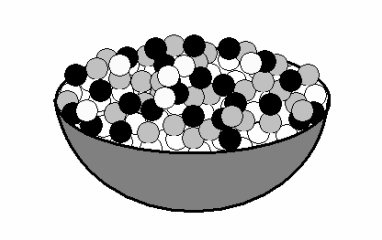

Figure 2.4.1: Urn with balls having different colors

We pick a ball at random and observe its color. The ball is put back in the urn and then a new ball is picked. Upon the completion of this experiment we have a sequence of balls that may have different colors. This is the simplest case of a random process. Now extend this concept to a more complex case.

Now suppose that instead of a single urn we have N urns. These urns contain balls having three colors with different proportions. In Figure 2.4.2, $N = 3$, and there are three colored balls, i.e., white, gray and black. We randomly select an urn according to some

probability distribution, and pick a ball from that urn. After noting the color of the ball, we place the ball in the same urn and then select another urn at random. The urn might be the same urn selected before or it can be a different urn.

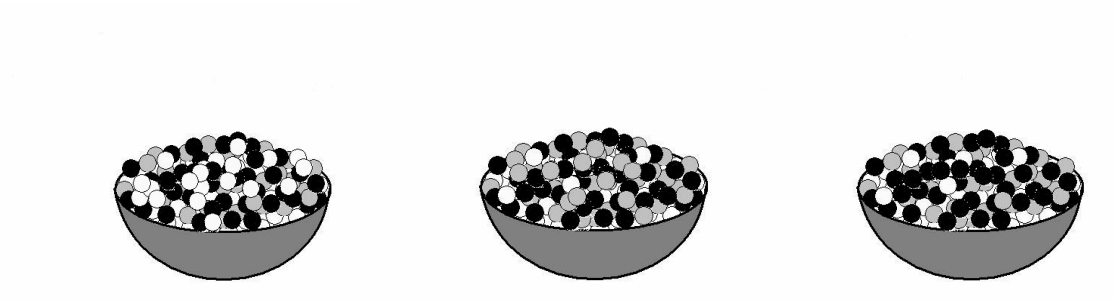

Figure 2.4.2: Three urns with balls having different colors

We pick a ball from the newly selected urn and observe its color. Hence there are two random processes occurring one after another, i.e., the selection of an urn and the selection of a ball from that particular urn.

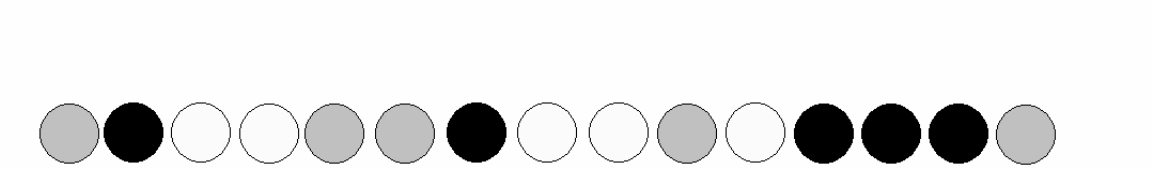

Figure 2.4.3: Sequence of balls drawn from the three urns

Figure 2.4.3 shows one possible sequence that can be obtained from this procedure. Note that every urn has in it a specific distribution for different colored balls. The urns are represented by states, and looking at the sequence shown in Figure 2.4.3, the sequence in which different urns are selected cannot be determined. We can only observe the color of the balls picked from the urns. We have no way of knowing the urns from which the balls were picked. This is an example of a *hidden* Markov model (HMM). The output probability distribution of each state may be different from each other and the probability of picking a ball depends upon the probability distribution of the balls in that particular urn (state of a Markov model).

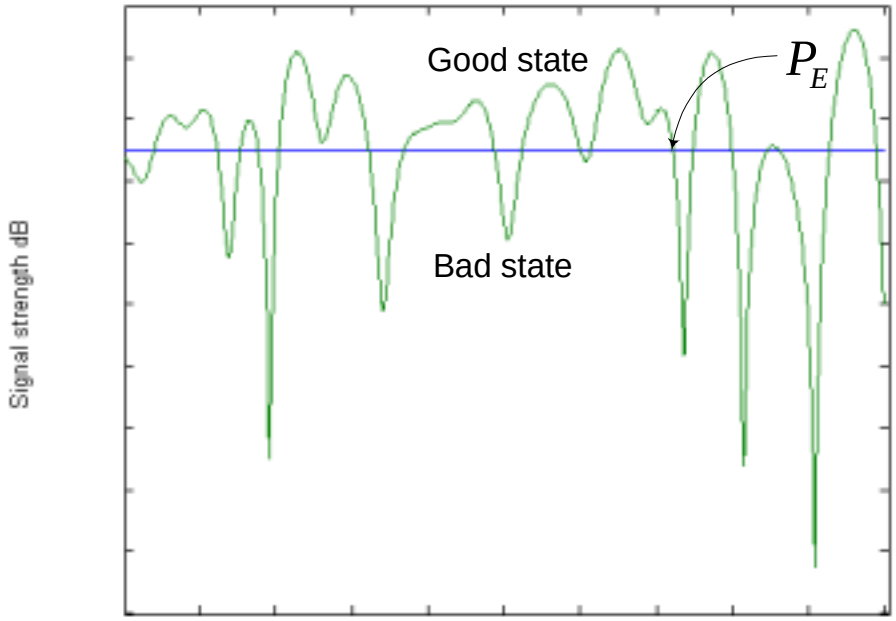

Figure 2.4.4: Illustration of a good and bad state in wireless channels

Let us now turn our attention to the realization of wireless channels. Consider the simplest case of a channel having only one state, and observe the errors generated by the channel. The generation of an error depends upon a threshold level P_E, the threshold being the probability of generating error. If the probability is less than the threshold, the error probability can be neglected. If the probability is higher than the threshold, then it is assumed that the error is generated. This is illustrated in Figure 2.4.4.

Now consider a more complex case where the channel consists of more than one state. Each state can have a different threshold level, and depending upon a particular threshold, we declare that whether a state is good or bad state, and associate an error probability with that state.

Many textbooks are devoted to the explanation of HMMs and people have been applying these models in a number of applications. Almost all of the references discuss these models in detail. See [1, 2, 6, 22, 52, 55, 57, 60 and 69].

2.5 Finite State Channel (FSC)[1]

Consider $\mathbf{a} = \{\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_m\}$ as the input to the channel, and the output of the channel is $\mathbf{b} = \{\mathbf{b}_1, \mathbf{b}_2, ..., \mathbf{b}_m\}$, where m is a non-negative integer $(m \in \mathrm{N})$. If the output **b** is same as the input **a,** except for the delay, then the channel is said to be *ideal* or *noiseless*. The channel impairments distort the transmitted symbols and the received sequence contains errors whose values are defined by some measure of difference between transmitted and received symbols. The channel distortion is usually characterized by the conditional probabilities

$$\Pr(\mathbf{b}_i, \mathbf{b}_{i+1}, ..., \mathbf{b}_{i+m} \mid \mathbf{a}_i, \mathbf{a}_{i+1}, ..., \mathbf{a}_{i+m}) \tag{2.5.1}$$

of receiving the sequence of symbols $\mathbf{b}_i, \mathbf{b}_{i+1}, ..., \mathbf{b}_{i+m}$ when the sequence $\mathbf{a}_i, \mathbf{a}_{i+1}, ..., \mathbf{a}_{i+m}$ was sent. The channel under consideration may also be described by the conditional probabilities of errors $\Pr(\mathbf{b}_i, \mathbf{b}_{i+1}, ..., \mathbf{b}_{i+m} \mid \mathbf{a}_i, \mathbf{a}_{i+1}, ..., \mathbf{a}_{i+m})$. If the error sequence $\mathbf{e}_i, \mathbf{e}_{i+1}, ..., \mathbf{e}_{i+m}$ does not depend upon the transmitted sequence

$$\Pr(\mathbf{e}_i, \mathbf{e}_{i+1}, ..., \mathbf{e}_{i+m} \mid \mathbf{a}_i, \mathbf{a}_{i+1}, ..., \mathbf{a}_{i+m}) = \Pr(\mathbf{e}_i, \mathbf{e}_{i+1}, ..., \mathbf{e}_{i+m}) \tag{2.5.2}$$

then the channel is said to be *symmetric*. The channel is called *stationary* if these probabilities do not depend on i. The complete description of the errors can be specified by the multidimensional distributions of errors $\Pr(\mathbf{e}_0, \mathbf{e}_1, ..., \mathbf{e}_m)$. The practical application of multidimensional distribution is quite limited because of their computational complexity. This is the reason for using an approximate channel error sequence description based on an error source model.

[1] Refer to [6] for more details.

Assume that the channel consists of the state space $S = \{s_1, s_2, ..., s_m\}$. Sometimes the states are enumerated by just integers. If at time $t-1$, the channel is in state s_{t-1} and the input to the channel is $\mathbf{a}_t \in A$, the channel outputs symbol $\mathbf{b}_t \in B$, and transfers to state $s_t \in S$ with probability $\Pr(\mathbf{b}_t, s_t \mid \mathbf{a}_t, s_{t-1})$. The probability of the final state s_t and receiving a sequence $\Pr(\mathbf{b}_t, s_t = j \mid \mathbf{a}_t, s_{t-1} = i)$ conditioned on the initial state s_0 and transmitted sequence $\mathbf{a}_1^t = (\mathbf{a}_1, \mathbf{a}_2, ..., \mathbf{a}_t)$ has the form

$$\Pr(\mathbf{b}_1^t, s_t \mid \mathbf{a}_1^t, s_0) = \sum_{s_1^{t-1}} \prod_{i=1}^{t} \Pr(\mathbf{b}_i, s_i \mid \mathbf{a}_i, s_{i-1}) \tag{2.5.3}$$

Writing the above equation in the matrix form, define $\Pr(\mathbf{b}_t \mid \mathbf{a}_t)$ is defined as a conditional matrix probability (MP) of the output symbol given the input symbol as a matrix whose *ij*-th element is $\Pr(\mathbf{b}_t, s_t = j \mid \mathbf{a}_t, s_{t-1} = i)$

$$\mathbf{P}(\mathbf{b}_t \mid \mathbf{a}_t) = \left[\Pr(\mathbf{b}_t, s_t \mid \mathbf{a}_t, s_{t-1})\right]_{u,u} \tag{2.5.4}$$

The conditional probability $\Pr(\mathbf{b}_1^t \mid \mathbf{a}_1^t, s_0)$ of receiving $\mathbf{b}_1^t$ conditioned on the channel initial state s_0 and the input sequence $\mathbf{a}_1^t$ is the sum over s_t of $\Pr(\mathbf{b}_1^t, s_t \mid \mathbf{a}_1^t, s_0)$. The conditional probability $\Pr(\mathbf{b}_1^t \mid \mathbf{a}_1^t)$ of receiving $\mathbf{b}_1^t$ conditional on the channel input sequence $\mathbf{a}_1^t$ is given by

$$\Pr(\mathbf{b}_1^t \mid \mathbf{a}_1^t) = \sum_{s_0=1}^{u} \Pr(s_0) \Pr(\mathbf{b}_1^t \mid \mathbf{a}_1^t, s_0) \tag{2.5.5}$$

or $$\Pr(\mathbf{b}_1^t \mid \mathbf{a}_1^t) = \mathbf{p}_0 \mathrm{p}(\mathbf{b}_1^t \mid \mathbf{a}_1^t) = \mathbf{p}_0 \mathbf{P}(\mathbf{b}_1^t \mid \mathbf{a}_1^t)\mathbf{1} = \mathbf{p}_0 \prod_{i=1}^{t} \mathbf{P}(\mathbf{b}_i \mid \mathbf{a}_i)\mathbf{1} \tag{2.5.6}$$

where $\mathbf{p}_0 = \left[\Pr(s_0 = 1) \;\; \Pr(s_0 = 2) \; ... \; \Pr(s_0 = u)\right]$ is the matrix of the channel initial state probabilities.

The autonomous FSC is the channel where output does not depend upon the inputs. This source model is sometimes called a *finite state generator* (FSG), and the FSC is called the

transducer. The source is described by A, state space S, matrix row $\mathbf{p}_s$ of the state initial probabilities, and the matrix probability $\mathbf{P}_s(\mathbf{a})$ whose elements are the probabilities $\Pr(\mathbf{a}, j \mid i)$ of transferring from the source state i to state j and producing the symbol $\mathbf{a}$. The probability of producing a sequence $\mathbf{a}_1^t$ by the FSG is given by

$$\Pr(\mathbf{a}_1^t) = \mathbf{p}_s \prod_{t=1}^{t} \mathbf{P}_s(\mathbf{a}_i)\mathbf{1} \tag{2.5.7}$$

where the elements of symbol MP $\mathbf{P}_s(\mathbf{a})$ can be written as

$$\Pr(\mathbf{a}, j \mid i) = \Pr(j \mid i) \Pr(\mathbf{a} \mid i, j) \tag{2.5.8}$$

This constitutes a Markov chain with the transition matrix

$$\mathbf{P} = \left[\Pr(j \mid i)\right] = \sum_{\mathbf{a}} \mathrm{P}_s(\mathbf{a}) \tag{2.5.9}$$

The model states are not directly observable, which means that the state cannot be identified by observing the symbol. The state sequence is a probabilistic function of the hidden states, and the symbol conditional probability, given a sequence of states, depends only on the previous and the current states. This probability depends only on the current state $\Pr(\mathbf{a}_t \mid i, j) = \Pr(\mathbf{a}_t \mid j)$, whose model is called a *discrete* hidden Markov model (HMM). Hence we can say that *a hidden Markov model (HMM) is a doubly embedded stochastic process with an underlying stochastic process that is not directly observable, but can be observed through a set of stochastic processes that produce the sequence of observations.* This model can be used to address the problem of the channel input (source) modeling.

In the majority of applications, it is assumed that the underlying Markov chains are stationary, and if the matrix $\mathbf{P}$ is regular, then the state initial probability vector can be found as a solution of the system

$$\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi} \qquad \boldsymbol{\pi}\mathbf{1} = 1 \tag{2.5.10}$$

2.6 Gilbert-Elliott's Model

The Gilbert-Elliott model [4, 24, 38] is a time varying binary symmetric channel. The crossover probabilities of the channel are determined by the current state of a discrete-time stationary binary Markov process. The source has two states: G (for good or no errors) and B (for bad or burst errors). This constitutes the Markov chain with the state transition matrix

$$\mathbf{P} = \begin{bmatrix} Q & p \\ P & q \end{bmatrix} \tag{2.6.1}$$

The probability of going back to good state, given the model is in good state, is Q and the probability going to bad state is p. The probability of going back to bad state and the probability of going from bad state to good state is q and P respectively. The diagrammatic view of this process is shown below.

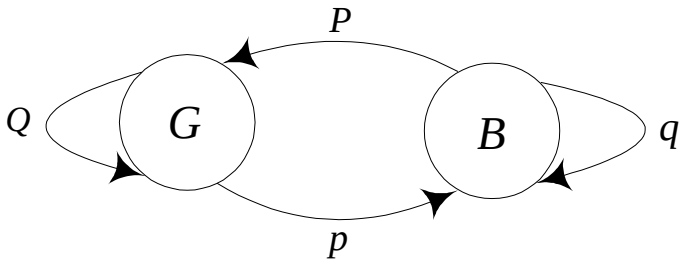

Figure 2.6.1: The Gilbert-Elliott's model

Figure 2.6.2 shows the expanded Gilbert's model state diagram. There are two bad states B_1 and B_2, and a single good state G. Turin in [6] and [69] discussed this model in detail with some examples. This is the basic model for the illustration of HMM in bursty wireless channels [4, 12, 17, 19, 21, 33, 37, 38, and 43]. Note that Gilbert's basic model is not enough for modeling wireless channels in most cases. Hence we use Gilbert's expanded model for modeling these channels more accurately. Flow graph methods [6]

and [69] are also useful in understanding the Markovian processes for Gilbert-Elliott channels.

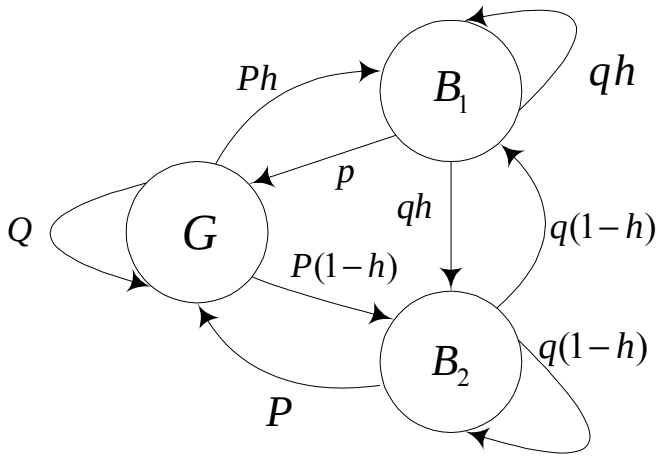

Figure 2.6.2: Expanded Gilbert model with one good and two bad states

2.7 Fritchman's Partitioned Markov Chains

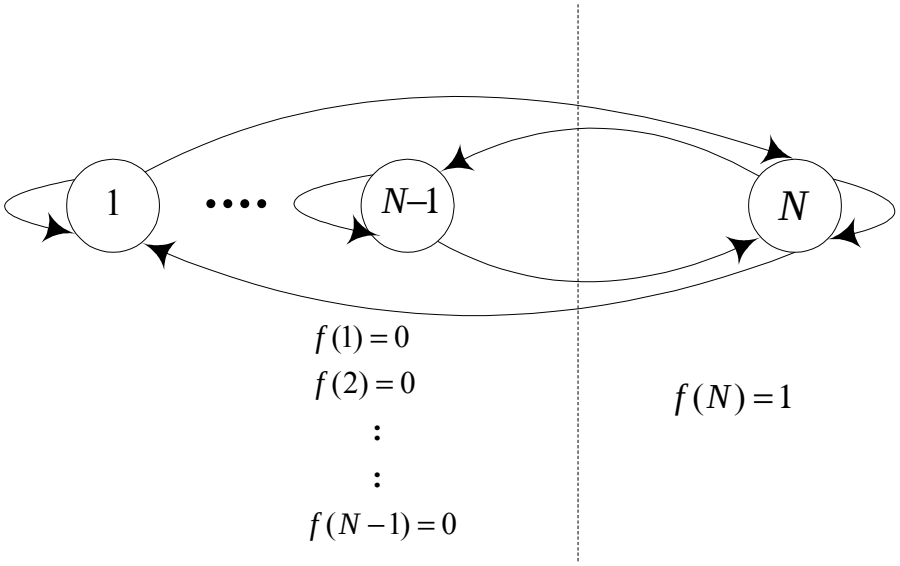

Figure 2.7.1: Fritchman's model

Fritchman [5] studied the Markov chains where the outputs are a deterministic function of the states. This is the basic idea for semi-hidden Markov models [2], and these models will be discussed in more detail in the later sections of this thesis. We use partitioned matrix for particular observations in this case.

Figure 2.7.1 shows a semi-hidden Markov model with N states and having one bad state. This is Fritchman's channel with $(N-1)$ good and only one bad state. Good states generate only zeroes (no errors) and bad state generates only ones (errors). The number of good or bad states can be increased depending on the nature of the error patterns.

2.8 The Error Probability

The error probability can be obtained from the parameters of the Markov models without reproducing the sequence. This can be achieved by raising the state transition matrix $\mathbf{P}$ to a very high power. In other words, we compute

$$\mathbf{\Phi} = \mathbf{P}^{\infty} \tag{2.8.1}$$

$\mathbf{\Phi}$ is known as the *limiting multistep* transition probability matrix of the process. Its rows give the probability assignment over the states of the process that must exist for each starting state when the number of transitions of the process are allowed to increase to a large number. We find out that the rows of the $\mathbf{\Phi}$ matrix are identical; the ultimate probability assignment does not depend where the process started. This matrix shows the state transition probability of the stationary Markov process. Multiplying the probability of transitions of states into itself with the corresponding error probability of the states gives the error probability of the resulting error sequence.

If $\mathbf{B}$ denotes the output symbol probability matrix, with the second row denoting the probability of generating error in corresponding states, then the error probability for an N state Markov chain can be written as

$$P_E = \sum_{i=1}^{N} \mathbf{\Phi}(i,i)\,\mathbf{B}(2,i) \tag{2.8.2}$$

where P_E denotes the probability of error. This shows that the error probability of the Markov models can be directly obtained from the model parameters without reproducing the error sequences and calculating their corresponding bit errors. [1, 6 and 69] discuss the limiting behavior of Markov chains in considerable detail.

Example 2.8.1

The state transition matrix for a certain hidden Markov model is given as follows

$$\mathbf{P} = \begin{bmatrix} 0.8115 & 0.8892 & 0.9406 \\ 0.0112 & 0.9830 & 0.0058 \\ 0.0088 & 0.0762 & 0.9150 \end{bmatrix}$$

and the output symbol probability matrix $\mathbf{B}$ is given by

$$\mathbf{B} = \begin{bmatrix} 0.9959 & 0.8892 & 0.9406 \\ 0.0041 & 0.1108 & 0.0594 \end{bmatrix}$$

which shows that there are two outputs in the process (errors and no errors). Making the state transition matrix invariant by raising it to a very high power gives

$$\mathbf{\Phi} = \mathbf{P}^{\infty} = \begin{bmatrix} 0.0553 & 0.8773 & 0.0674 \\ 0.0553 & 0.8773 & 0.0674 \\ 0.0553 & 0.8773 & 0.0674 \end{bmatrix}$$

The error probability from (2.8.3) is given as

$$P_E = \mathbf{\Phi}(1,1)\mathbf{B}(2,1) + \mathbf{\Phi}(2,2)\mathbf{B}(2,2) + \mathbf{\Phi}(3,3)\mathbf{B}(2,3)$$

$$= 0.1015$$

Note that in this example the second row of the $\mathbf{B}$ matrix gives the probability of generating an error.

2.9 State Duration

This section tells us how to determine the probability of being in state for τ consecutive instants. Let $p_i(\tau)$ be the probability density function defined as

$$p_i(\tau) = \Pr(q_1 = s_i, q_2 = s_i, q_3 = s_i, ..., q_\tau = s_i, q_{\tau+1} = s_j, ...) \tag{2.9.1}$$

$$= \pi_i (a_{ii})^{\tau-1}(1-a_{ii}) \tag{2.9.2}$$

Now determine the expected duration in state i by the following equation

$$\overline{\tau}_i = \sum_{\tau=1}^{\infty} \tau p_i(\tau) \tag{2.9.3}$$

Hence

$$\overline{\tau}_i = \sum_{\tau=1}^{\infty} \tau (a_{ii})^{\tau-1}(1-a_{ii}) \tag{2.9.4}$$

$$= (1-a_{ii}) \sum_{\tau=1}^{\infty} \tau (a_{ii})^{\tau-1} \tag{2.9.5}$$

$$= (1-a_{ii}) \frac{\partial}{\partial a_{ii}} \sum_{\tau=1}^{\infty} \tau (a_{ii})^{\tau-1} \tag{2.9.6}$$

$$= (1-a_{ii}) \frac{\partial}{\partial a_{ii}} \left(\frac{a_{ii}}{1-a_{ii}} \right) \tag{2.9.7}$$

$$= \frac{1}{1-a_{ii}} \tag{2.9.8}$$

Example 2.9.1

Suppose that there is a three state Markov process in which state 1 denotes *Rainy* (R), state 2 denotes *Cloudy* (C) and state 3 denotes *Sunny* (S) weather. The state transition probability matrix is given as

$$\mathbf{P} = \begin{bmatrix} 0.4 & 0.3 & 0.3 \\ 0.2 & 0.6 & 0.2 \\ 0.1 & 0.1 & 0.8 \end{bmatrix}$$

The average consecutive sunny, cloudy and rainy days are given as

$$\text{Rainy days} = \frac{1}{1-a_{11}} = \frac{1}{1-0.4} = 1.7$$

$$\text{Cloudy days} = \frac{1}{1-a_{22}} = \frac{1}{1-0.6} = 2.5$$

$$\text{Sunny days} = \frac{1}{1-a_{33}} = \frac{1}{1-0.8} = 5$$

Note however that the duration distribution is normally meaningless in many physical systems.

2.10 Semi-hidden Markov Processes

Error source modeling can be generalized using semi-hidden Markov processes. These are sometimes also referred to as semi-Markov processes or *Markov renewal processes* [2, 6, 69, 12 and 21]. The semi-Markov approach often leads to a significant reduction of the model parameter set and hence computationally efficient models can be achieved as compared with HMM in most cases. We can view of semi-Markov process as the process whose successive state occupancies are governed by the transition probabilities of a Markov process, but whose stay in any state is described by an integer-valued random variable that depends on the state presently occupied, and on the state to which the next transition will be made. Thus at transition instants the semi-Markov process behaves just like a Markov process. This process is also called the embedded Markov process. However, the time at which transition occurs is governed by a different probabilistic mechanism. The semi-Markov process can be described mathematically as follows.

Let p_{ij} be the probability that a semi-Markov process that entered state i on its last transition will enter j on its next transition. The transition probabilities, p_{ij}, must satisfy the same equations as the transition probabilities for a Markov process,

$$p_{ij} \geq 0 \qquad i = 1,2,...,N; \quad j = 1,2,...,N \tag{2.10.1}$$

and

$$\sum_{j=1}^{N} p_{ij} = 1 \qquad i = 1,2,...,N \tag{2.10.2}$$

where the process has N number of states. Whenever the system is in state i, it determines the next state j according to the state i's transition probabilities $p_{i1}, p_{i2}, ..., p_{iN}$. However, after j has been selected, but before making this transition from state i to state j, the process *holds* for a time τ_{ij}, in state i. The holding time, τ_{ij}, is a positive, integer-valued random variable governed by a probability mass function $h_{ij}(.)$, known as the holding time mass function for a transition from state i to state j. Thus,

$$\Pr(\tau_{ij} = m) = h_{ij}(m) \qquad m = 1,2,3,...; i, j = 1,2,...,N; \tag{2.10.3}$$

Assume that the means $\bar{\tau}_{ij}$ of all holding time distributions are finite and that all holding time mass distributions are at least one time unit in length,

$$h_{ij}(0) = 0 \tag{2.10.4}$$

Consider that a set of states and symbols are generated by a set of sources. Given a symbol, we cannot say which state s generates this particular symbol. The model states are not observable, and hence we will call this model a semi-hidden Markov model (SHMM or semi-HMM). For a particular case of error source modeling, these models are general enough for modeling a wide variety of error sources.

Thus for the particular case of error source modeling, semi-Markov models can be described with the matrix of initial state probabilities, the matrix of transition probabilities, the matrices of interval distributions and the matrices of probabilities of error. The described error source model is known as the semi-Markov model, and the corresponding channel is called a *symmetric* semi-Markov channel. If the interval distribution of a semi-Markov process depends only on the current state

$$h_{ij}(m) = h_i(m) \qquad i = 1,2,...,N; \; j = 1,2,...,N \tag{2.10.5}$$

then the process is called *autonomous*.

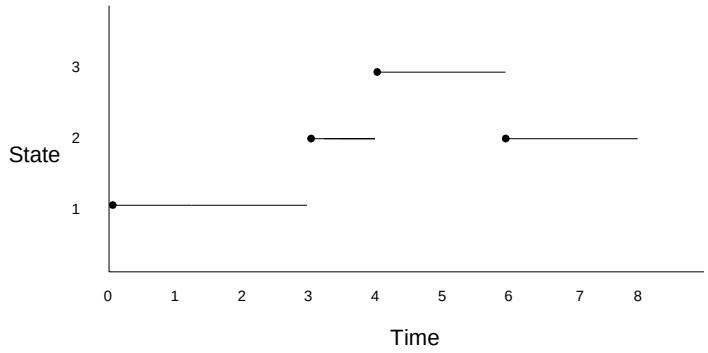

Figure 2.10.1: Semi-Markov process

Figure (2.10.1) shows the portion of a possible trajectory for a discrete-time semi-Markov process. The large black dots show that the process enters state immediately after a transition. The figure shows that the process entered the state 1 at time 0, was held in state 1 for 3 time units, and then made the transition to state 2. It then held in state 2 for 1 time unit, and then jumped to state 3. It held in state 3 for 2 time units, and then jumped to state 2.

Note that the basic connection between the time scale and the time of transition is removed. We can talk about the probability of three transitions in a total time, etc. We can also observe that the discrete-time Markov process can be considered to be a discrete-time semi-Markov process for which

$$h_{ij}(m) = \delta(m-1) \qquad m = 0,1,2,...; i, j = 1,2,...,N; \tag{2.10.6}$$

where δ denotes the delta function and N is the total number of states of the Markov process. This shows that the holding times are exactly one time unit in length.

2.10.1 Real and Virtual Transitions

Real transitions are the transitions that require an actual change to state indices. Virtual transitions are the transitions where the state indices could be the same after the transition. We can view a Markov process as a process that can make only real transitions

and have geometric holding time distributions, or as a process that can make virtual transitions and have all holding time equal to unit one. This shows that a discrete- time Markov process is a discrete-time semi-Markov process.

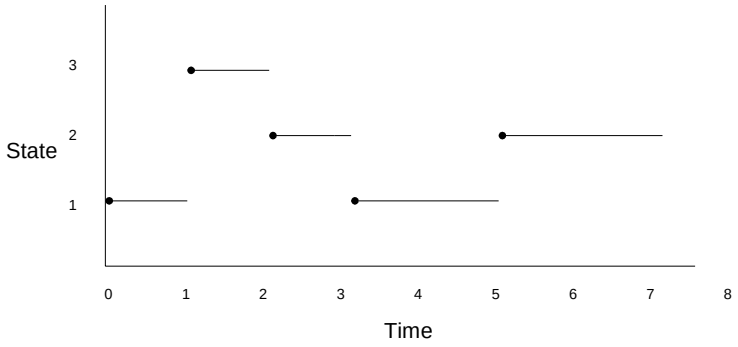

Figure 2.10.1.1: Real transitions

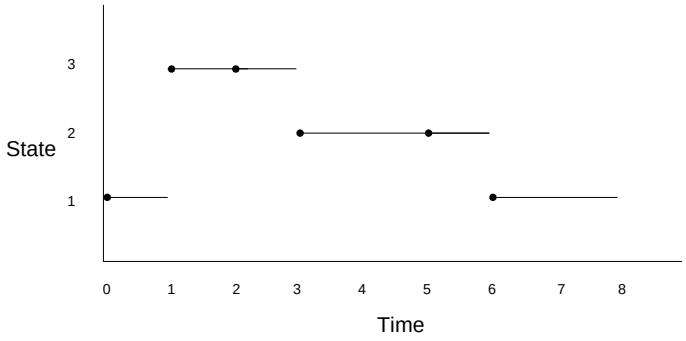

Figure 2.10.1.2: Virtual transitions

The decision to call a movement from a given state back to the same state depends upon the circumstances. Some physical systems require that only real transitions be allowed, in other processes virtual transitions are the most important. We observe that when virtual transitions are allowed, transitions can occur with no change of state. Figures 2.10.1.1 and 2.10.1.2 show the graphical illustration of real and virtual transitions respectively.

2.10.2 Semi-Markov Modeling of Error Sources

The traditional Baum-Welch algorithm (BWA), which will be discussed in Chapter 3, is widely used for estimating parameters of error sources in real communication channels. These algorithms sometimes require a prohibitive number of computations, because the intervals between two consecutive bursts could be quite large. For the case where there are very low BERs, the mean durations become very large and the observation sequences can be very long. We exploit this property and try to do modifications in existing algorithms so that the computational efficiency and the speed of convergence can be increased. The basic idea of this model is given by Fritchman [5]. Refer to [6, 69, 12, and 37] for further detail.

A sequence of zeros and ones 00001100000000000000111000000011000000 can be compactly written as $0^4 1^2 0^{14} 1^3 0^6 1^2 0^5$. This method of compression is called *runlength coding*.

The main advantage of using this notation in algorithms is that fast matrix exponentiation and analytical expression can be used to estimate the parameters of Markov models. The trade off between computational efficiency and speed is discussed for our particular case of modeling error sources for the W-CDMA environment using real world channel measurements from VIPER [72, 73].

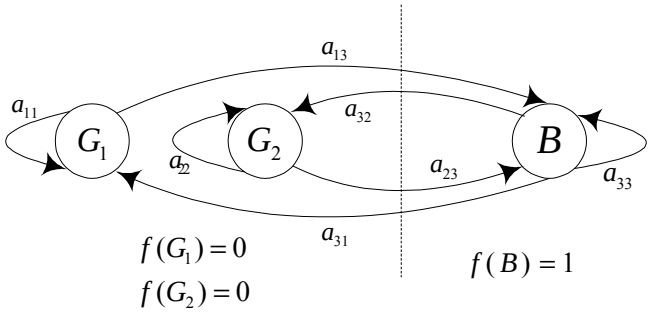

Figure 2.10.2.1: Semi-Markov models

$$\mathbf{P} = \begin{bmatrix} a_{11} & 0 & a_{13} \\ 0 & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad (2.10.2.1)$$

Figure 2.10.2.1 shows the error source modeling for three states, with two good states and one bad state. The state transition matrix of the model is also shown in (2.10.2.1). For a model with two good states and two bad states, the state transition matrix could take the form shown in (2.10.2.2).

$$\mathbf{P} = \begin{bmatrix} a_{11} & 0 & a_{13} & a_{14} \\ 0 & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & 0 \\ a_{41} & a_{42} & 0 & a_{44} \end{bmatrix} \tag{2.10.2.2}$$

2.11 Block Matrix Representation

It is sometimes convenient to represent a matrix in the form of blocks. It will be observed that block matrices can be used to estimate the parameters for semi-hidden Markov models. Consider a Markov process that produces two observation symbols. There are a total of N states, and the states can be partitioned into two subsets, N_0 and N_1, with N_0 representing the states that produces the observation symbol 0, and the subset N_1 contains the states that produce the observation symbol 1. In other words, we can attribute all the states with 0 output symbol to state N_0, and correspondingly N_1 to the states with observation symbol 1. We can also attribute all the states with the initial number of $i = 1, 2, ..., R$ to the subset N_0, and the remaining states with the numbers $i = R+1, R+2, ..., N$ attributed to the subset N_1. The state transition matrix $\mathbf{P}$ can be written as

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{00} & \mathbf{P}_{01} \\ \mathbf{P}_{10} & \mathbf{P}_{11} \end{bmatrix} \tag{2.11.1}$$

Using the approach in error source modeling, we say that no error occurs when the Markov process is in state N_0, and error occurs when the process is in state N_1. Hence

$$\mathbf{P}(0) = \begin{bmatrix} \mathbf{P}_{00} & 0 \\ \mathbf{P}_{10} & 0 \end{bmatrix} \qquad \mathbf{P}(1) = \begin{bmatrix} 0 & \mathbf{P}_{01} \\ 0 & \mathbf{P}_{11} \end{bmatrix} \tag{2.11.2}$$

Notice that the observation symbol from a particular state is a deterministic function of that state and hence this model is a Markov function. This representation is widely used in error source modeling using semi-Markov models. The mathematical representation of models using this approach is not usually compact, but since the rank of matrices are smaller, we get faster computation.

2.12 Regeneration of Model Specific Sequences

In this section we first review the properties of the matrices that are obtained from the Markov models estimation process. These are:

$\boldsymbol{\pi}$, $1 \times N$ initial state probability vector. This vector helps us in determining the initial state of the channel. This matrix is used only once during the regeneration of error sequences.

$\mathbf{P}$, $N \times N$ state transition matrix with elements (p_{ij}). This matrix gives the probability of being in a state, provided that the previous state is known.

$\mathbf{B}$, $L \times N$ output symbol probability matrix with elements $(b_{ij}) = (b_j(k))$. This matrix gives us the probability of generating all the possible L outputs from a particular state. This is the matrix that differentiates HMM from ordinary Markov processes. In our case of error source simulation, L is generally equal to 2.

The error sequences can be obtained from the following way.

1. Start from the $\boldsymbol{\pi}$ matrix and select the initial state by generating a uniformly distributed random number. Select the initial state according to the probability distribution of the $\boldsymbol{\pi}$ matrix.

2. The output symbol is obtained by again generating a uniformly distributed random number and then assigning an output symbol according to the probability distribution of $\mathbf{B}$ the matrix. The $\mathbf{B}$ matrix tells us the output symbol from the given state.

3. The next state is then obtained by again generating a uniformly distributed random number and selecting the next state according to the probability distribution of the previous state from the $\mathbf{P}$ matrix.

4. Go back to process 2 and continue till the required number of the output symbols are obtained.

In the case of semi-Markov models, the $\mathbf{B}$ matrix is a deterministic function of the output. In the case of error source modeling, the output can take two values, either 0 or 1. In this case, it has 2 rows and N columns.

2.13 Interval Distributions

Let $\boldsymbol{x} = \{x_1, x_2, ..., x_T\}$ be an error sequence, with $x_k = 1$ indicating that the $k-$th transmitted bit suffered a transmitted error, and $x_k = 0$ indicates an error free transmission. Also, let the notation $(0^m \mid 1)$ denote the event of observing m or more consecutive (i.e. burst of) error free transmissions following an error, and $(1^m \mid 0)$ represents the event of observing m or more consecutive errors following a good (no error) period. Fritchman showed that the probabilities of occurrence of these two events are given by the sum of weighted exponentials

$$\Pr(0^m \mid 1) = \sum_{i=1}^{k} f_i \lambda_i^{m-1} \tag{2.13.1}$$

and

$$\Pr(1^m \mid 0) = \sum_{i=k+1}^{N} f_i \lambda_i^{m-1} \tag{2.13.2}$$

where $\lambda_{1,2,3,\ldots,k}, \lambda_{k+1,k+2,\ldots,N}$ are the eigenvalues of $\mathbf{A}_{GG}$ and $\mathbf{A}_{BB}$ respectively, and f_i's are function of a_{ij}'s. We can obtain the probability of obtaining exactly m zeros (or ones) as

$$\Pr(0^{m-1} \mid 1) - \Pr(0^m \mid 1) = \sum_{i=1}^{k} f_i \lambda_i^{m-2}(1-\lambda_i) \tag{2.13.3}$$

$$\Pr(1^{m-1} \mid 0) - \Pr(1^m \mid 0) = \sum_{i=k+1}^{N} f_i \lambda_i^{m-2}(1-\lambda_i) \tag{2.13.4}$$

The Fritchman model can be interpreted as equivalent to a Markov process with a state transition probability matrix

$$\mathbf{A} = \begin{bmatrix} \mathbf{\Lambda}_{GG} & \mathbf{A}_{GB} \\ \mathbf{A}_{BG} & \mathbf{\Lambda}_{BB} \end{bmatrix} \tag{2.13.5}$$

$$\mathbf{\Lambda}_{GG} = \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_k \end{bmatrix} \tag{2.13.6}$$

$$\mathbf{\Lambda}_{BB} = \begin{bmatrix} \lambda_{k+1} & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \tag{2.13.7}$$

Note that in this equivalent model there are no transitions within the set of k good states, and no transitions within the set of $N-k$ bad states. Such transitions are indistinguishable from the observed error sequence, since a transition from one good state to another good state still produces no errors, and the transition is not observable from the point.

The Fritchman model is not unique except when there is only one bad state [21]. This is so because, in the general case, the error-free run distribution $\Pr(0^m \mid 1)$ and the error burst

distribution $\Pr(1^m \mid 0)$ do not specify the statistical dependence of the error free runs and the error bursts. In the case of a single error state model, Fritchman showed that

$$\Pr(0^m \mid 1) = \sum_{k=1}^{N-1} \frac{a_{Nk}(a_{kk})^m}{a_{kk}} \tag{2.13.8}$$

2.14 Fading Wireless Channels

We observe the process of fitting Markov models to different wireless channel scenarios [10, 44, 70 and 71]. These models are general enough to fit in different fading wireless channels. These fading scenarios include channels with and without multipath. The simulation results are shown for both flat fading and frequency selective channels [44].

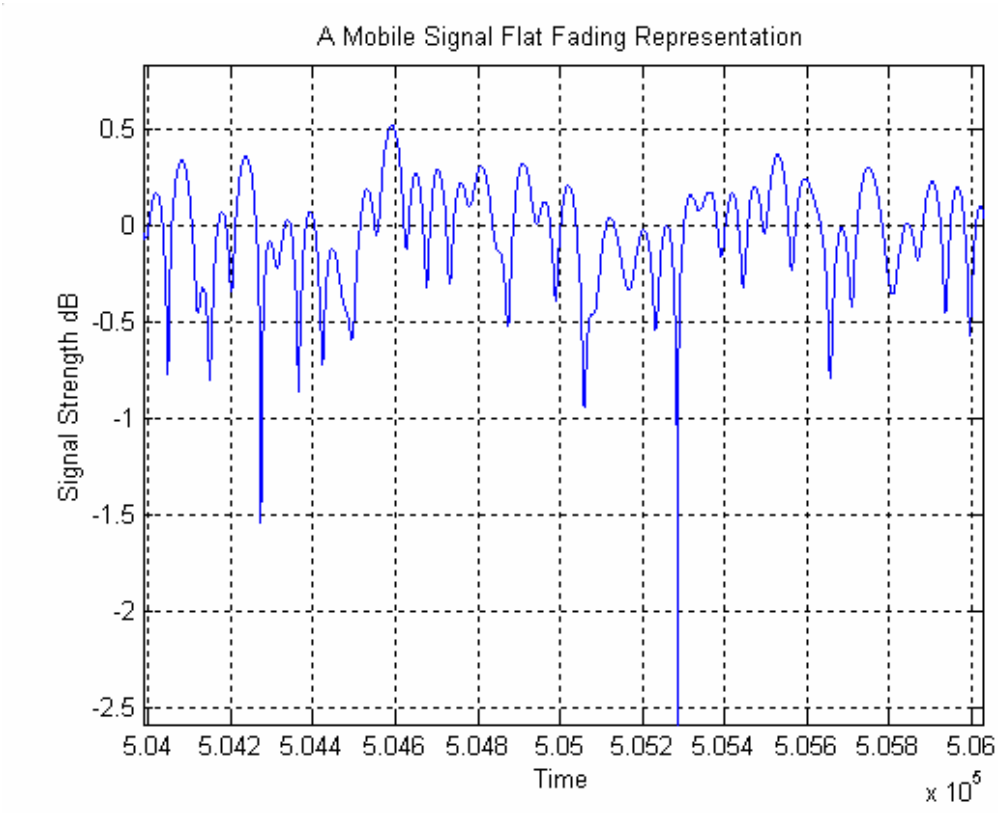

Figure 2.14.1: Typical Rayleigh fading envelope

Wideband code division multiple access (W-CDMA) channels with Vector ImPulsE Response (VIPER) channel measurements [73] are observed for Markov chains. Wireless

channels can generate complex error patterns especially when we have multipath. For a flat fading channel, Clark's model [44, 45, 68, 70 and 71] is used. The model states that flat fading envelopes follow Rayleigh distribution.

The wireless channel is a complex system to model, especially when there are many interferes and multipath. Figure 2.14.1 shows the fading envelope of a typical Rayleigh channel and Figure 2.14.2 shows its corresponding probability density function.

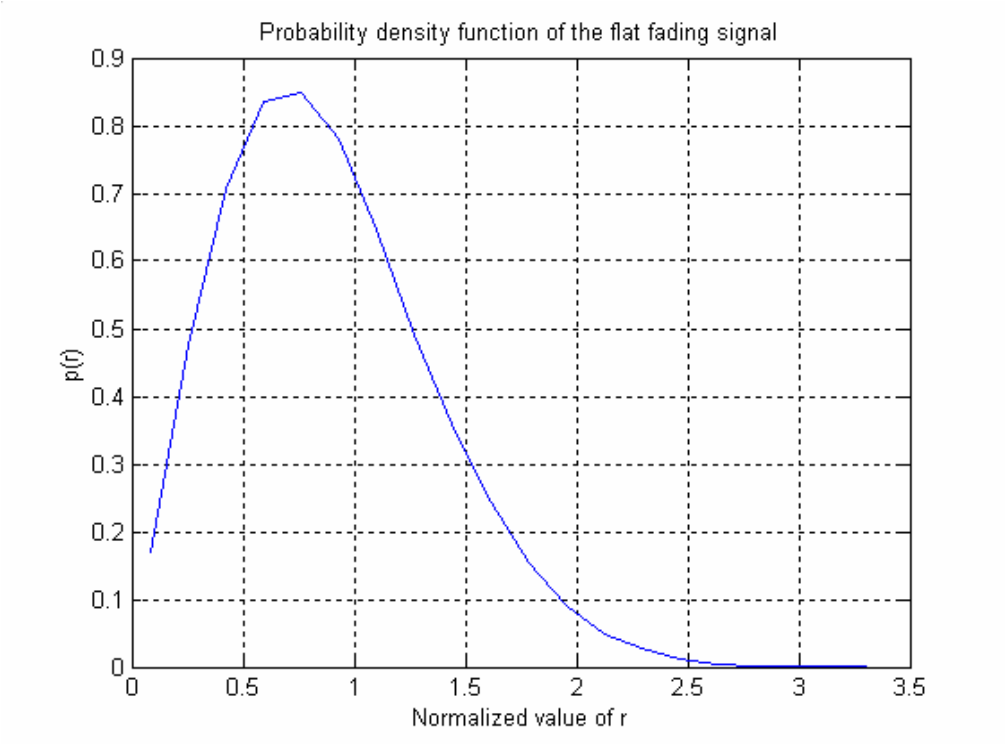

Figure 2.14.2: Probability density function of the Rayleigh fading envelope

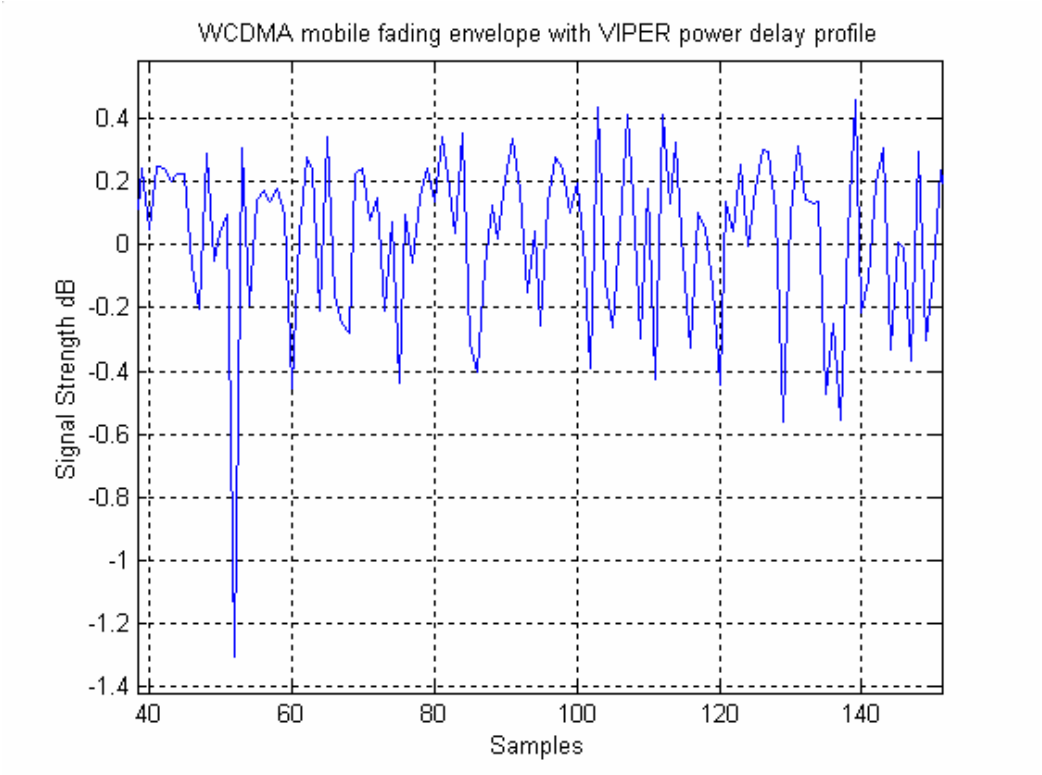

Figure 2.14.3: Fading envelope of the W-CDMA environment

Figure 2.14.3 shows the fading envelope obtained from the W-CDMA simulator with power delay profiles taken from VIPER.

2.15 Summary

This chapter summarizes the basic ideas related to hidden Markov and semi-hidden Markov models. Mathematical descriptions of Markov models and their realization for wireless channels are discussed in considerable detail. We will discuss the issue of parameter estimation of these models (both hidden Markov models and semi-hidden Markov models) in more detail.

Chapter

3

Parameter Estimation

This chapter discusses different methods of estimating the parameters of Markov models. We start with the Kullback-Leibler Divergence (KLD), and then two well-known algorithms, the minimization algorithm and the Expectation-Maximization (EM) algorithm, are introduced. In this chapter, a modified form of the forward-only semi-Markov model is also introduced. This model can estimate the parameter of a Markov model using runlength coding and does not use the backward path. Most of this work is taken from [6] and some concepts are further illustrated in detail. Simulation results using these models, especially for W-CDMA channels with real world measurements taken from VIPER, are discussed in Chapter 5.

3.1 Kullback-Leibler Divergence (KLD)

The Kullback-Leibler Divergence (KLD) [75, and 77] deals with approximating a stochastic process with some distribution. This divergence is also known as *relative entropy* and is a measure of the difference between two probability distributions. This divergence criterion plays a major role in obtaining different algorithms that are used to estimate the parameters of Markov models. By using this theory we select or discard a certain model.

Let us now use mathematical expressions to explain this divergence criterion. Consider a stochastic process x with the cumulative probability distribution $f(x)$. We will try to approximate this stochastic process with a model whose probability distribution is given as $g(x,\tau)$. The approximation quality measure $L(\tau)$ is shown below

$$L(\tau) = \Delta(f(x), g(x,\tau)) \tag{3.1.1}$$

and we try to minimize this measure. The small the measure, the closer it is to the model with the stochastic process. The optimal fit is achieved by minimizing $L(\tau)$ with respect to τ such that

$$\hat{\tau} = arg \min_{\tau} L(\tau) \quad \tau \in \Omega \tag{3.1.2}$$

Two quality measures that we use most frequently are [6]

$$\chi^2(\tau) = \sum_x \left[f(x) - g(x,\tau) \right]^2 / g(x,\tau) \tag{3.1.3}$$

and the Kullback-Leibler Divergence (KLD) [75]

$$D(f \,||\, g_\tau) = \sum_x f(x) \log(f(x)/g(x,\tau)) = E\{\log(f(x)/g(x,\tau) \,|\, f(x)\} \tag{3.1.4}$$

where E denotes the expectation.

3.2 Minimization Algorithm

We need to solve the minimization problem described by (3.1.1) in order to find the optimal fit. It is almost impossible to solve this problem without iterations, and hence the minimum is found iteratively. We find the sequence of parameters $\tau_0, \tau_1, \tau_2, \ldots$ that decreases the function $L(\tau)$. This forms the basis for the algorithms that iteratively find the parameters of HMMs. The first-order approximation is

$$L(\tau) \approx L(\tau_p) + \frac{\partial L(\tau_p)}{\partial \tau_p} (\tau - \tau_p) \tag{3.2.1}$$

where τ is a column vector and the gradient is a row vector. The direction of the function's fastest decrease at the point τ_p is opposite to its gradient $\partial L(\tau_p)/\partial\tau_p$. Thus we can write

$$\tau'_{p+1} = \tau'_p - \lambda_p \frac{\partial L(\tau_p)}{\partial \tau_p} \qquad p = 0, 1, 2, ... \tag{3.2.2}$$

where p denotes the iterations. If λ_p is sufficiently small, then $L(\tau_p) \geq L(\tau_{p+1})$, and hence a closer approximation is obtained. This shows that the quality of measure increases after every iteration, but the problem is to find an optimal λ_p.

The second-order approximation is given as

$$L(\tau) \approx L(\tau_p) + \frac{\partial L(\tau_p)}{\partial \tau_p}(\tau - \tau_p) + (\tau - \tau_p)' \frac{\partial^2 L(\tau_p)}{\partial \tau_p^2}(\tau - \tau_p) \tag{3.2.3}$$

We can develop an iterative algorithm by approximating $L(\tau)$ with some function $q(\tau, \tau_p)$ with respect to τ

$$\tau_{p+1} = arg \min_{\tau} q(\tau, \tau_p) \tag{3.2.4}$$

This function minimizes $q(\tau, \tau_p)$ with respect to τ with $L(\tau_p) \geq L(\tau_{p+1})$. Define the residual $h(\tau, \tau_p)$ as

$$h(\tau, \tau_p) = L(\tau) - q(\tau, \tau_p) \tag{3.2.5}$$

or

$$L(\tau) = q(\tau, \tau_p) + h(\tau, \tau_p) \tag{3.2.6}$$

The residual $h(\tau, \tau_p)$ is much smaller than $q(\tau, \tau_p)$ for good approximations. We cannot guarantee that $L(\tau)$ decreases whenever $q(\tau, \tau_p)$ decreases. On the other hand, if both $q(\tau, \tau_p)$ and $h(\tau, \tau_p)$ decrease, then $L(\tau)$ also decreases. Moreover, if $h(\tau, \tau_p)$ always decreases (in other word, if $h(\tau_p, \tau_p)$ is its global maximum)

$$h(\tau,\tau_p) \le h(\tau_p,\tau_p) \qquad \forall \tau \in \Omega \tag{3.2.7}$$

then a decrease of $q(\tau,\tau_p)$ causes a decrease of $L(\tau)$. Now the expectation-maximization (EM) algorithm [74] will be discussed, which is a special case of this algorithm.

3.3 The Expectation-Maximization (EM) algorithm

The EM algorithm [74] estimates the parameters of a model iteratively, starting from some initial guess. Each iterations is an *Expectation* step, given the known values for the observed variables and the current estimate of the parameters, and a *Maximization* step, which re-estimates the parameters to be those with maximum likelihood, under the assumption that the distribution found in the Expectation step is correct. It can be shown that the distribution found in this step is correct. It can be also shown that each such iteration improves the true likelihood, or leaves it unchanged (if a local maximum has already been reached, or in uncommon cases, before then). For detailed mathematical representation and illustration of this algorithm, refer to [6, 69, and 74].

Consider the likelihood function [7]

$$L(\tau) = \sum_x f(x) \log g(x,\tau) = E\{\log g(x,\tau) \mid f(x)\} \tag{3.3.1}$$

Suppose that there is a function $\kappa(z;x,\tau) > 0$ whose sum over z is a constant. We can always assume that the constant is equal to one by normalizing this function. This can be shown as

$$\sum_z \kappa(z;x,\tau) = 1 \tag{3.3.2}$$

and then this expression can also be interpreted as a conditional probability of z. Denote

$$\Psi(z;x,\tau) = g(x,\tau)\,\kappa(z;x,\tau) \tag{3.3.3}$$

or

$$\log g(x,\tau) = \log \Psi(z;x,\tau) - \log \kappa(z;x,\tau) \tag{3.3.4}$$

multiplying both sides of the equation with $r(z;x,\tau) = f(x)\kappa(z;x,\tau)$ and summing with respect to z and x yields

$$\sum_x f(x)\log g(x,\tau) = \sum_x \sum_z \log\Psi(z;x,\tau)\, r(z;x,\tau_p) - \sum_x \sum_z \log\kappa(z;x,\tau)\, r(z;x,\tau_p) \qquad (3.3.5)$$

if we assume $\sum_z \kappa(z;x,\tau) = 1$, then (3.3.5) can be written as

$$L(\tau) = q(\tau,\tau_p) + h(\tau,\tau_p) \qquad (3.3.6)$$

where

$$q(\tau,\tau_p) = \sum_x \sum_z \log\Psi(z;x,\tau)\, r(z;x,\tau_p) = E\{\log\Psi(z;x,\tau)\,|\,r(z;x,\tau_p)\} \qquad (3.3.7)$$

$$h(\tau,\tau_p) = -\sum_x \sum_z \log\kappa(z;x,\tau)\, r(z;x,\tau_p) = -E\{\log\kappa(z;x,\tau)\,|\,r(z;x,\tau_p)\} \qquad (3.3.8)$$

We know that

$$E\{\log\kappa(z;x,\tau)\mid\kappa(z;x,\tau_p)\} \le E\{\log\kappa(z;x,\tau_p)\mid\kappa(z;x,\tau_p)\} \qquad (3.3.9)$$

and multiplying both the sides of this inequality by $f(x)$ and summing, we obtain $h(\tau,\tau_p) \ge h(\tau_p,\tau_p)$, which proves that $h(\tau_p,\tau_p)$ is a global minimum. So the likelihood algorithm can be found using

$$\tau_{p+1} = arg\min_\tau q(\tau,\tau_p) \qquad (3.3.10)$$

In summary,

$$\hat{\tau} = arg\max_\tau E\{\log g(x,\tau)\mid f(x)\} \qquad (3.3.11)$$

Iteratively by using algorithm

$$\tau_{p+1} = arg\max_\tau E\{\log\Psi(z;x,\tau)\,|\,r(z;x,\tau_p)\} \qquad p = 0,1,2,... \qquad (3.3.12)$$

and with each iteration increase $E\{\log g(x,\tau)\,|\,f(x)\}$. Every iteration consists of two steps:

1. *Expectation*: Calculating $E\{\log \Psi(z;x,\tau)\,|\,r(z;x,\tau_p)\}$
2. *Maximization*: Finding a maximum of $E\{\log \Psi(z;x,\tau)\,|\,r(z;x,\tau_p)\}$

The first step of this algorithm significantly increases the likelihood $L(\tau)$, and the subsequent steps deliver smaller increments. $L(\tau)$ is monotonically increasing and is bounded by $E\{\log f(x)\,|\,f(x)\}$, as it converges to some value L^*. The KLD, which is the difference between $L(\tau_p)$ and the upper bound $E\{\log f(x)\,|\,f(x)\}$, gives us an idea of the approximation quality. The limiting point could be a local maxima or a saddle point. The saddle point, however, is not stable, and small rounding errors usually throw us on the right track. The number of iterations should not be decided on the basis of the function change but rather on the value of the KLD.

3.4 Phase-Type (PH) Distribution Approximation

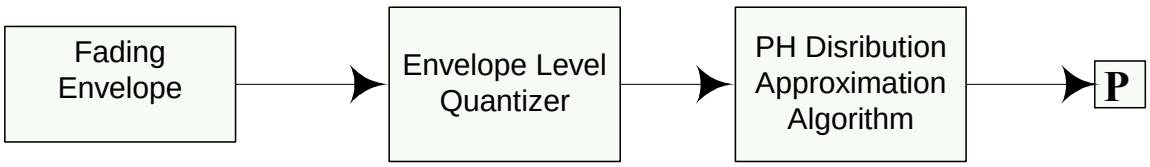

Figure 3.4.1: Phase-Type distribution approximation flow graph

Phase-Type distribution [6] or PH distribution is used to estimate the fade duration [44, 70] occurring in wireless channels. In Chapter 5, the simulation results for both flat fading and frequency selective channels will be shown.

Phase-Type distribution is a special case of matrix-geometric distribution with the form

$$g(x,\tau) = \boldsymbol{\pi}\,\mathbf{A}^{x-1}\,\mathbf{b} \tag{3.4.1}$$

where $\boldsymbol{\pi}$ is a row vector, $\mathbf{b}$ is a column vector, and $\mathbf{A}$ is a $u \times u$ square matrix.

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \boldsymbol{\pi} & 0 \end{bmatrix} \tag{3.4.2}$$

is a stochastic matrix. By definition,

$$\boldsymbol{\pi}\mathbf{1} = 1 \tag{3.4.3}$$

$$\mathbf{b} = (\mathbf{I} - \mathbf{A})\mathbf{1} \tag{3.4.4}$$

In this section we will develop an EM algorithm for approximating a probability distribution $f(x)$ with phase-type (PH) distribution.

To develop the EM algorithm, the matrix product is written as

$$g(x, \tau) = \sum_{z} \pi_{i_0} \, a_{i_0 i_1} \ldots a_{i_{x-2} i_{x-1}} a_{i_{x-1}, u+1} \tag{3.4.5}$$

where $a_{i,u+1} = b_i$ and $z = \boldsymbol{i}_0^{x-1}$. Define

$$\psi(z; x, \tau) = \pi_{i_0} \, a_{i_0 i_1} \ldots a_{i_{x-2} i_{x-1}} \, a_{i_{x-1}, u+1} \tag{3.4.6}$$

so that

$$g(x, \tau) = \sum_{z} \psi(z; \tau) \tag{3.4.7}$$

Dividing both sides of this equation by $g(x, \tau)$, and defining $\kappa(z; x, \tau) = \psi(z; x, \tau) / g(x, \tau)$, it can be rewritten that

$$\sum_{z} \kappa(z; x, \tau) = 1 \tag{3.4.8}$$

In this way $\kappa(z; x, \tau)$ can be interpreted as a probability distribution.

Define $\rho(x, \tau_p) = f(x) / g(x, \tau)$, and replace the $\log \psi(z; x, \tau)$ as the sum of logarithms

$$\log \psi(z; x, \tau) = \log \tau_{i_0} + \sum_{k=1}^{x} \log a_{i_{k-1}, i_k} \tag{3.4.9}$$

Note that $$\sum_{z,i_0=i} \psi(z;x,\tau) = \sum_{i_1^{x-1}} a_{i,i_1}\, a_{i_1,i_2} \ldots a_{i_{x-2},i_{x-1}}\, a_{i_{x-1},u+1} \tag{3.4.10}$$

is the $i-$th element of the matrix

$$\boldsymbol{\beta}(x,\tau)=\mathbf{A}^{x-1}\mathbf{b} \tag{3.4.11}$$

Since $$\sum_{i=1}^{u} \pi_i \beta_i(x,\tau) = \boldsymbol{\pi}\boldsymbol{\beta}(x,\tau) = \boldsymbol{\pi}\,\mathbf{A}^{x-1}\mathbf{b} = g(x,\tau) \tag{3.4.12}$$

we have $$\sum_x \rho(x,\tau)\, g(x,\tau) = \sum_x f(x) = 1 \tag{3.4.13}$$

and hence

$$\pi_{i,p+1} = \pi_{i,p} \sum_x \beta_i(x,\tau_p)\rho(x,\tau_p) \tag{3.4.14}$$

This result can be used in simulations directly.

Referring to [6], the BWA algorithm for fitting the PH distribution to $f(x)$ is given as follows

$$\pi_{i,p+1} = \pi_{i,p} \sum_x \rho(x,\tau_p)\beta_i(x,\tau_p) \tag{3.4.15}$$

$$a_{ij,p+1} = a_{ij,p} \frac{\sum_x \rho(x,\tau_p) \sum_{k=1}^{x-1} \alpha_i(k-1,\tau_p)\beta_j(x-k,\tau_p)}{\sum_x \rho(x,\tau_p) \sum_{k=1}^{x} \alpha_i(k-1,\tau_p)\beta_i(x-k+1,\tau_p)} \tag{3.4.16}$$

$$b_{i,p+1} = b_{i,p} \frac{\sum_x \rho(x,\tau_p)\, \alpha_i(x-1,\tau_p)}{\sum_x \rho(x,\tau_p) \sum_{k=1}^{x} \alpha_i(k-1,\tau_p)\beta_j(x-k+1,\tau_p)} \tag{3.4.17}$$

Define the recursive forward algorithm as

$$\boldsymbol{\alpha}(0,\tau) = \boldsymbol{\pi} \qquad \boldsymbol{\alpha}(x,\tau) = \boldsymbol{\alpha}(x-1,\tau)\,\mathbf{A} \qquad x = 1,2,\ldots \tag{3.4.18}$$

and $\boldsymbol{\beta}(x,\tau)$ can be evaluated recursively by the backward algorithm

$$\boldsymbol{\beta}(1,\tau)=\mathbf{b} \qquad \boldsymbol{\beta}(x+1,\tau)=\mathbf{A}\boldsymbol{\beta}(x,\tau) \qquad x=1,2,... \tag{3.4.19}$$

All the elements of the forward variable $\boldsymbol{\alpha}(1)$, $\boldsymbol{\alpha}(2)$, ... , $\boldsymbol{\alpha}(x-1)$ and the backward variable $\boldsymbol{\beta}(1)$, $\boldsymbol{\beta}(2)$, ..., $\boldsymbol{\beta}(x-1)$ do not need to be saved if the data is processed by defining a matrix

$$\mathbf{W}(x,\tau)=\sum_{k=1}^{x-1}\mathbf{A}^{x-k-1}\mathbf{b}\,\boldsymbol{\pi}\,\mathbf{A}^{k-1}=\sum_{k=1}^{x}\boldsymbol{\beta}(x-k,\tau)\,\boldsymbol{\alpha}(k-1,\tau) \tag{3.4.20}$$

$\boldsymbol{\beta}(x-k,\tau)\boldsymbol{\alpha}(k-1,\tau)$ is a square matrix whose $ij-\text{th}$ element is $\alpha_j(k-1,\tau)\beta_i(x-k,\tau)$. We can write

$$\mathbf{W}(x+1,\tau)=\sum_{k=1}^{x}\mathbf{A}^{x-k}\mathbf{b}\boldsymbol{\pi}\mathbf{A}^{k-1}=\mathbf{A}\mathbf{W}(x,\tau)+\mathbf{b}\,\boldsymbol{\alpha}(x-1,\tau) \tag{3.4.21}$$

Hence the following forward-only algorithm can be obtained

$$\boldsymbol{\alpha}(0,\tau)=\boldsymbol{\pi} \qquad \boldsymbol{\beta}(1,\tau)=\mathbf{b} \qquad \mathbf{W}(1,\tau)=\mathbf{0} \tag{3.4.22}$$

Then compute

$$A_{ij}(x,\tau_p)=a_{ij,p}w_{ji}(x,\tau_p) \tag{3.4.23}$$

and

$$\rho(x,\tau)=\frac{f(x)}{g(x,\tau)} \tag{3.4.24}$$

with

$$g(x,\tau)\ =\boldsymbol{\pi}\boldsymbol{\beta}(x,\tau) \tag{3.4.25}$$

Update the sum of the estimated parameters using (3.4.15), (3.4.16) and (3.4.17).

Note that $\alpha_j(k-1,\tau)\beta_i(x-k,\tau)$ can be replaced with $A_{ij}(x,\tau)$, and this result can be easily used in simulations to estimate $a_{ij,p+1}$.

Recalculate

$$\begin{aligned}
&\mathbf{W}(k+1,\tau) = \mathbf{A}\mathbf{W}(k,\tau) + \mathbf{b}\,\boldsymbol{\alpha}(k-1,\tau) \\
&\boldsymbol{\alpha}(k,\tau) = \boldsymbol{\alpha}(k-1,\tau)\,\mathbf{A} \quad k = 1,2,...,x-1 \\
&\boldsymbol{\beta}(k+1,\tau) = \mathbf{A}\,\boldsymbol{\beta}(k,\tau)
\end{aligned} \tag{3.4.26}$$

and hence estimate the model parameters as

$$\begin{aligned}
a_{ij,p+1} &= C_i \sum_{m=1}^{T} A_{ij}(x_m, \tau_p) / g(x_m, \tau_p) \\
b_{i,p+1} &= C_i \sum_{m=1}^{T} \alpha_i(x_m - 1, \tau_p) / g(x_m, \tau_p)
\end{aligned} \tag{3.4.27}$$

where C_i is the normalization factor given as

$$C_i^{-1} = \sum_{m=1}^{T} \left[A_{ij}(x_m, \tau_p) / g(x_m, \tau_p) + \alpha_i(x_m - 1, \tau_p) / g(x_m, \tau_p) \right] \tag{3.4.28}$$

The memory requirements for this algorithm are very small because there is no need to keep track of forward and backward paths.

3.5 Forward-Backward Hidden Markov Models

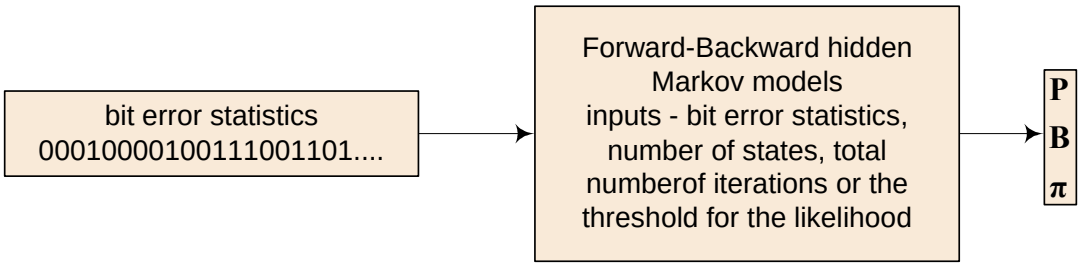

Figure 3.5.1: Forward-backward HMM flow graph

The Baum-Welch forward-backward algorithm [6, 11, 22 and 23] is the most efficient and prevalent method for automatically estimating the parameters of a set of hidden Markov models. Its biggest appeal lies in the way it effectively avoids the possible explosion in computational complexity resulting from the evaluation of all the states of a

model of training data. The key to the elimination of redundant computation is to exploit the fact that all possible state sequences must merge into one of the entire states set at one time.

The Baum-Welch algorithm (BWA) is a special case of EM algorithm. In this section, we will show how to use the previous theory to estimate of parameters of forward-backward BWA for forward-backward HMMs.

The implementation of the Baum-Welch algorithm involves computation of two different probability terms. First, the forward path probability is defined as the joint probability of having generated a partial observation sequence in the forward direction (i.e. from the start of the data), and having arrived at a certain state at a certain frame. Next, the backward path probability denotes the probability of generating a partial observation sequence in the reverse direction (from the final frame of data), given that the state sequence starts from a certain state at a certain time. The parameters of the HMMs can be estimated by using these probabilities. The full details of the Baum-Welch procedure for parameter estimation, as well as the various implementation issues, will be described in this and later sections.

In this section the representation of the Baum-Welch algorithm will be shown. Let us define a sequence $\boldsymbol{x}_1^T = x_1, x_2, ..., x_T$. Define the forward variable

$$\alpha_t(i) = \Pr(x_1, x_2, ..., x_t, q_t = s_i \mid \tau) \tag{3.5.1}$$

as the probability of the partial observation sequence $\{x_1, x_2, ..., x_t\}$ in state s_i at times step t given the model τ. This variable can be computed inductively, and from these variables, $\Pr(\boldsymbol{x} \mid \tau)$ is computed. The initialization is done as

$$\alpha_1(i, \tau) = \Pr(x_1, q_1 = s_i \mid \tau) = \Pr(x_1 \mid q_1 = s_i, \tau) \Pr(q_1 = s_i \mid \tau) \tag{3.5.2}$$

$$\alpha_1(i, \tau) = \pi_i b_i(x_1) \tag{3.5.3}$$

In the matrix form, the following equation can also be used

$$\boldsymbol{\alpha}(\boldsymbol{x}_1^0, \tau) = \boldsymbol{\pi} \tag{3.5.4}$$

and the induction is done as

$$\alpha_{t+1}(j) = \Pr(x_1, x_2, ..., x_{t+1}, q_{t+1} = s_j \mid \tau) \tag{3.5.5}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \Pr(x_1, x_2, ..., x_{t+1}, q_{t+1} = s_j \mid \tau) \Pr(q_{t+1} = s_j \mid q_t = s_i, \tau) \right] \Pr(x_{t+1} \mid q_{t+1} = s_j, \tau) \tag{3.5.6}$$

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^{N} \alpha_t(i)\, a_{ij} \right] b_j(x_{t+1}), \quad t \in \{1, 2, ..., T-1\},\ j \in \{1, 2, ..., N\} \tag{3.5.7}$$

If we denote $\alpha_i(\mathbf{x}_1^t, \tau)$ and as the elements of $\boldsymbol{\alpha}(\boldsymbol{x}_1^t, \tau)$,then in matrix form, it can be written as

$$\boldsymbol{\alpha}(\boldsymbol{x}_1^t, \tau) = \boldsymbol{\alpha}(\boldsymbol{x}_1^{t-1}, \tau)\mathbf{P}(x_t, \tau) \qquad t = 1, 2, ..., T \tag{3.5.8}$$

where $\mathbf{P}(x_t, \tau)$ is a function of the state transition matrix and the output symbol probability matrix, $\mathbf{b}$, for the current input sequence.

$\boldsymbol{\alpha}$ is a row vector and it needs to be *normalized*. The normalization can be done by dividing the elements of the forward variable by its sum for a particular time instant.

Figure 3.5.2 shows the graphical computation for $\boldsymbol{\alpha}$.

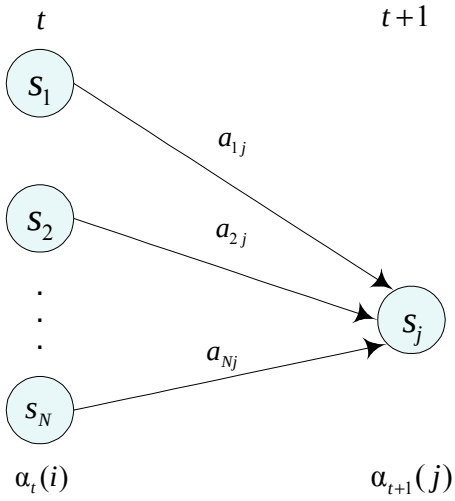

Figure 3.5.2: Computation of forward variable

Let us again define a backward variable $\beta_t(i)$ as

$$\beta_t(i) = \Pr(x_{t+1}, x_{t+2}, ..., x_T \mid q_t = s_i, \tau) \tag{3.5.9}$$

which denotes the probability of the partial observation $\{x_{t+1}, x_{t+2}, ..., x_T\}$ given that at time step t the model is in state s_i and given the model τ. The initialization is done

$$\beta_T(i) = 1, \qquad i \in \{1, 2, ..., N\} \tag{3.5.10}$$

and the matrix form is written as

$$\boldsymbol{\beta}(\boldsymbol{x}_T^t, \tau) = \mathbf{1} \tag{3.5.11}$$

The induction is done as

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij} b_j(x_{t+1}) \beta_{t+1}(j), \qquad t = \{T-1, T-2, ..., 1\}, i = \{1, 2, ..., N\} \tag{3.5.12}$$

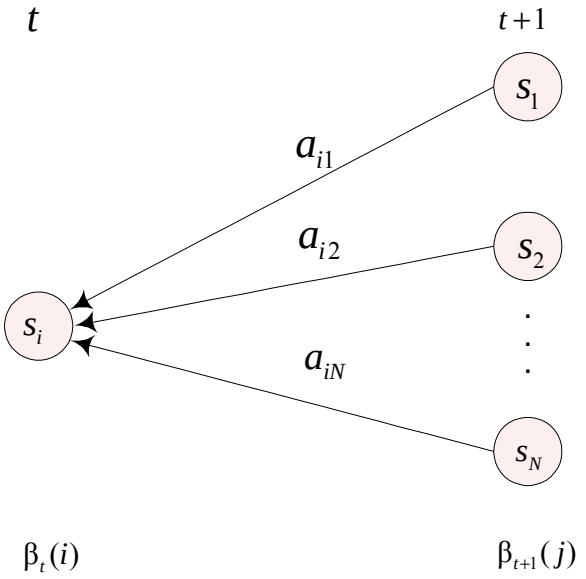

Figure 3.5.3: Computation of backward variable

If we denote $\beta_i(\boldsymbol{x}_1^t, \tau)$ as the elements of $\boldsymbol{\beta}(\boldsymbol{x}_1^t, \tau)$, then we can use the matrix form and write

$$\boldsymbol{\beta}(\boldsymbol{x}_t^T, \tau) = \mathbf{P}(x_t, \tau)\, \boldsymbol{\beta}(\boldsymbol{x}_{t+1}^T, \tau) \qquad t = T, T-1, ..., 1 \tag{3.5.13}$$

where $\mathbf{P}(x_t, \tau)$ is a function of the state transition matrix and the output symbol probability matrix $\mathbf{b}$ for the current input sequence. This backward variable is column vector and needs to be scaled by dividing the elements of the backward variable by the sum obtained from adding the forward variable for the corresponding time index. Scaling is discussed in the next section.

Note that

$$\Pr(\boldsymbol{x} \mid \tau) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \qquad t \in \{1, 2, ..., T\} \tag{3.5.14}$$

The estimated parameters of forward-backward BWA are given as

$$p_{ij} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i} \tag{3.5.15}$$

$$b_j(k) = \frac{\text{expected number of times in state } s_j \text{ and observing symbol } v_k}{\text{expected number of times in state } s_j} \tag{3.5.16}$$

$$\pi_i = \text{expected number of times in state } s_i \text{ at times } t = 1 \tag{3.5.17}$$

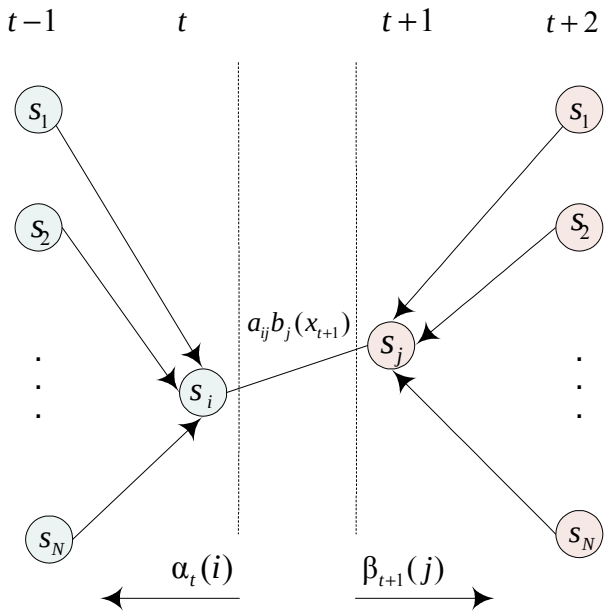

Figure 3.5.4: Parameter estimation

The probability of obtaining a sequence given the model is

$$\Pr(\boldsymbol{x} \mid \tau) = \sum_{i=1}^{N} \alpha_t(i)\beta_t(i) \qquad t \in \{1,2,...,T\} \tag{3.5.18}$$

The parameters are estimated as follows

$$p_{ij_{p+1}} = \frac{\sum_{t=1}^{T} \alpha_{t_p}(i)\, p_{ij_p} b_{j_p}(x_{t+1})\beta_{t+1_p}(j)}{\sum_{t=1}^{T} \alpha_{t_p}(i)\beta_{t_p}(i)} \tag{3.5.19}$$

$$b_{j_{p+1}}(k) = \frac{\sum_{\substack{t \\ \forall x_t = v_k}} \alpha_{t_p}(j)\beta_{t_p}(j)}{\sum_{t=1}^{T} \alpha_{t_p}(j)\beta_{t_p}(j)} \tag{3.5.20}$$

$$\pi_i = \alpha_1(i)\beta_1(i) \tag{3.5.21}$$

Appendix A explains this algorithm in further detail with the introduction of some intermediate variables. This is the basic algorithm and it converges slowly. This algorithm uses the sequence in a sample-by-sample basis and hence is not computationally efficient. Many modifications are proposed and some of these modifications are used here to increase the computational efficiency and to decrease the simulation runtime.

3.5.1 Scaling

Most of the papers dedicated to estimating the parameters of HMMs do not address the issue of scaling the Markov model parameters. The forward and backward vectors tend to zeros exponentially for large data size, if not scaled properly, and the machine suffers numerical underflow if these variables are not properly scaled. We can scale the variables if we define

$$\bar{\alpha}_t(j) = \frac{\alpha_t(j)}{\sum_{i=1}^{N} \alpha_t(i)} \tag{3.5.1.1}$$

and define the scaling vector C_t as

$$C_t = \sum_{i=1}^{N} \alpha_t(i) \tag{3.5.1.2}$$

This implies that

$$\sum_{i=1}^{N} \bar{\alpha}_t(i) = 1 \tag{3.5.1.3}$$

and

$$\bar{\beta}_t(i) = \frac{\beta_t(i)}{C_t} \tag{3.5.1.4}$$

with the initialization of

$$\bar{\beta}_T = \frac{\mathbf{1}}{C_T} \tag{3.5.1.5}$$

where $\mathbf{1}$ denotes the column vector containing all ones. This method avoids numerical underflow in these variables. The gamma variable can also be normalized but its normalization is not necessary. Refer to Turin [6] for more detail about scaling. Note that no scaling is required for estimating the parameters of PH distribution.

3.6 Forward-Only Hidden Markov Model

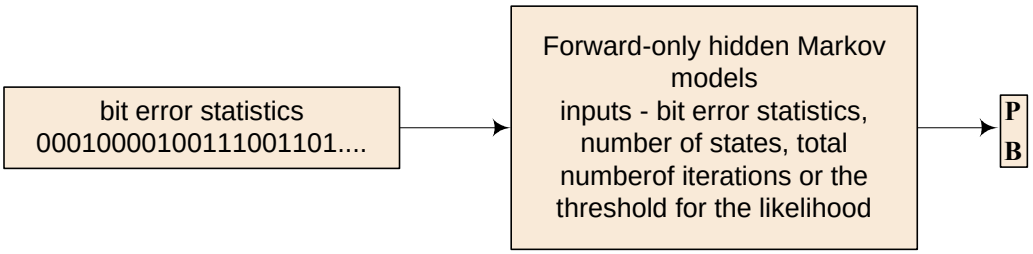

Figure 3.6.1: Forward-only HMM flow graph

In the forward-backward implementation, the algorithm requires the amount of memory that is proportional to the data size T, that can be very large. In contrast, unidirectional algorithms require the amount of memory that is independent of the data size and it also uses less memory than the forward-backward Baum-Welch algorithm, because the backward variable $\boldsymbol{\beta}$ is not used during the re-estimation. We only keep the forward variable in memory and this also makes the model easily programmable in a DSP chip. We can apply the forward-only Baum-Welch algorithm online without waiting until the whole sequence of length T has been received. For further detail, refer to [6]. We can also discard previous sequences, and then estimate the model from the estimated parameters obtained from those sequences. Let us now represent the mathematical expressions for estimating the parameters of the forward-only BWA.

From the forward-backward BWA, it is evident that we have to compute sums of the form

$$S(\boldsymbol{x}_1^T) = \sum_{t+1}^{T} \boldsymbol{\alpha}(\boldsymbol{x}_1^T)\mathbf{U}(t, x_t)\boldsymbol{\beta}(\boldsymbol{x}_{t+1}^T) \tag{3.6.1}$$

where $\mathbf{U}(t, x_t)$ is a matrix. We will try to find efficient ways of representing these matrices. In forward-backward BWA the standard form for determining these matrices are used.

It follows from the forward and backward probability definitions that

$$S = \boldsymbol{\pi}\ \mathbf{S}(\boldsymbol{x}_1^T)\mathbf{1} \tag{3.6.2}$$

where

$$\mathbf{S}(\boldsymbol{x}_1^T) = \sum_{t=1}^{T} \mathbf{P}(\boldsymbol{x}_1^t)\mathbf{U}(t, x_t)\mathbf{P}(\boldsymbol{x}_{t+1}^T) \tag{3.6.3}$$

and

$$\mathbf{P}(\boldsymbol{x}_u^v) = \prod_{i=u}^{v} \mathbf{P}(x_i) \tag{3.6.4}$$

are the corresponding matrix probabilities.

It can be written that for $u \le v \le w$

$$\mathbf{P}(\boldsymbol{x}_u^w) = \mathbf{P}(\boldsymbol{x}_u^v)\mathbf{P}(\boldsymbol{x}_v^w) \tag{3.6.5}$$

$$\mathbf{S}(\boldsymbol{x}_u^w) = \mathbf{S}(\boldsymbol{x}_u^v)\mathbf{P}(\boldsymbol{x}_{v+1}^w) + \mathbf{P}(\boldsymbol{x}_u^v)\mathbf{S}(\boldsymbol{x}_{v+1}^w) \tag{3.6.6}$$

These equations form the basis for the recursive algorithm.

Remember that the forward variable in the forward-backward BWA is computed using

$$\boldsymbol{\alpha}(\boldsymbol{x}_1^{t_{k+1}}) = \boldsymbol{\alpha}(\boldsymbol{x}_1^{t_k})\mathbf{P}(\boldsymbol{x}_{t_k+1}^{t_{k+1}}) \tag{3.6.7}$$

If we define

$$\boldsymbol{s}(\boldsymbol{x}_1^{t_{k+1}}) = \boldsymbol{s}(\boldsymbol{x}_1^{t_k})\mathbf{P}(x_{t_k+1}^{t_{k+1}}) + \boldsymbol{\alpha}(\boldsymbol{x}_1^{t_k})\mathbf{P}(x_{t_k+1}^{t_{k+1}}) \tag{3.6.8}$$

where

$$\boldsymbol{s}(\boldsymbol{x}_1^t) = \boldsymbol{\pi}\,\mathbf{S}(\boldsymbol{x}_1^t) \tag{3.6.9}$$

and if $t_k = t$ and $t_{k+1} = t_k$, we have the following

$$\boldsymbol{\alpha}(\boldsymbol{x}_1^{t+1}) = \boldsymbol{\alpha}(\boldsymbol{x}_1^t)\mathbf{P}(\boldsymbol{x}_{t+1}) \tag{3.6.10}$$

$$\boldsymbol{s}(\boldsymbol{x}_1^{t+1}) = \boldsymbol{s}(\boldsymbol{x}_1^t)\mathbf{P}(x_{t+1}) + \boldsymbol{\alpha}(\boldsymbol{x}_1^t)\mathbf{U}(t, x_{t+1}) \tag{3.6.11}$$

The numerator of the equation

$$p_{ij_{p+1}} = \frac{\sum_{t=1}^{T} \alpha_{t_p}(i)\, p_{ij_p} b_{j_p}(x_{t+1}) \beta_{t+1_p}(j)}{\sum_{t=1}^{T} \alpha_{t_p}(i) \beta_{t_p}(i)} \tag{3.6.12}$$

can be written as

$$S(\boldsymbol{x}_1^T) = \sum_{t=1}^{T} \boldsymbol{\alpha}(\boldsymbol{x}_1^{t-1}) \mathbf{U}_{ij}\, p_{ij} b_{j,p}(x_t) \boldsymbol{\beta}(\boldsymbol{x}_{t+1}^T) \tag{3.6.13}$$

where $\mathbf{U}_{ij}$ is the $N \times N$ matrix whose ij - th element is one, and the rest of its elements are zeros. It follows from this equation that

$$\boldsymbol{s}_{ij}(\boldsymbol{x}_1^{t+1}) = \boldsymbol{s}_{ij}(\boldsymbol{x}_1^t) \mathbf{P}(x_{t+1}) + \boldsymbol{\alpha}(\boldsymbol{x}_1^t) \mathbf{U}_{ij}\, p_{ij,p} b_{j,p}(x_t) \tag{3.6.14}$$

The numerator of equation

$$b_{j_{p+1}}(k) = \frac{\sum_{\substack{t \\ \forall x_t = v_k}} \alpha_{t_p}(j) \beta_{t_p}(j)}{\sum_{t=1}^{T} \alpha_{t_p}(j) \beta_{t_p}(j)} \tag{3.6.15}$$

can be written as $$S_j(x) = \sum_{t=1}^{T} \boldsymbol{\alpha}(\boldsymbol{x}_1^t) \mathbf{U}_{jj} \delta(x, x_t) \boldsymbol{\beta}(\boldsymbol{x}_{t+1}^T) \tag{3.6.16}$$

where $\delta(x, x_t) = 1$, if $x = x_t$, and $\delta(x, x_t) = 0$ if $x \neq x_t$. Hence we can write

$$\mathbf{s}_j(\boldsymbol{x}_1^{t+1}, x) = \mathbf{s}_j(\boldsymbol{x}_1^t, x) \mathbf{P}(x_{t+1}) + \boldsymbol{\alpha}(\boldsymbol{x}_1^t) \delta(x, x_{t+1}) \mathbf{U}_{ii} \tag{3.6.17}$$

Also

$$\mathbf{s}_{ij}(\boldsymbol{x}_1^{t+1}) = \mathbf{s}_{ij}(\boldsymbol{x}_1^t) \mathbf{P}(x_{t+1}) + \boldsymbol{\alpha}(\boldsymbol{x}_1^t) \mathbf{U}_{ij}\, p_{ij,p} b_{j,p}(x_t) \tag{3.6.18}$$

$$\mathbf{s}_j(\boldsymbol{x}_1^{t+1}, x) = \mathbf{s}_j(\boldsymbol{x}_1^t, x) \mathbf{P}(x_{t+1}) + \boldsymbol{\alpha}(\boldsymbol{x}_1^t) \delta(x, x_{t+1}) \mathbf{U}_{ii} \tag{3.6.19}$$

are the equations that can replace forward-backward BWA.

Note that we do not have to keep any of these variables in memory in the form of an array. The above two equations represent the forward-only BWA. The initial conditions for the algorithm are

$$\boldsymbol{s}_{ij}(\boldsymbol{x}_1^0) = \mathbf{0} \tag{3.6.20}$$

$$\boldsymbol{s}_j(\boldsymbol{x}_1^0, x) = \mathbf{0} \tag{3.6.21}$$

The parameters of this forward-only BWA are estimated as

$$p_{ij,p+1} = \frac{\boldsymbol{s}_{ij}(\boldsymbol{x}_1^T)\mathbf{1}}{\sum_j \boldsymbol{s}_{ij}(\boldsymbol{x}_1^T)\mathbf{1}} \tag{3.6.21}$$

$$b_{j,p+1}(x) = \frac{\sum_j \boldsymbol{s}_j(\boldsymbol{x}_1^T)\mathbf{1}}{\sum_j \mathbf{b}_{j,p+1}(x)} \tag{3.6.22}$$

Note that by using only the $\boldsymbol{\alpha}$ variable for parameter estimations and not using any backward path reduces the memory requirements, but it might sometime give sub-optimal results. In Chapter 5 we will show the simulation results used in this algorithm.

3.7 Forward-Backward Semi-Markov Models [6, 12]

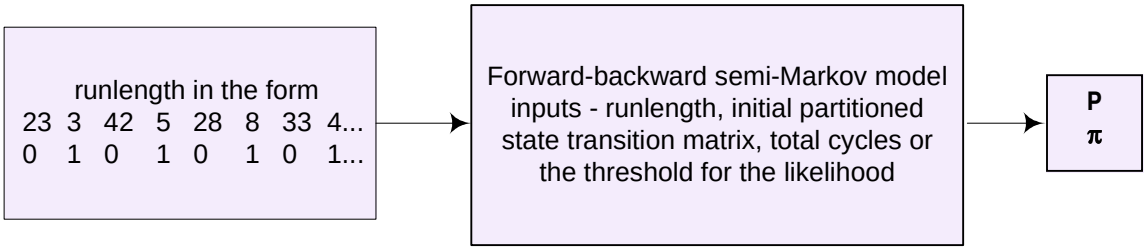

Figure 3.7.1: Forward-backward semi-Markov model flow graph

Consider a Markov model that follows the Markov function. Hence the set of states can be partitioned into subsets corresponding to the events, and the state transition matrix takes the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \dots & \mathbf{P}_{1s} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \dots & \mathbf{P}_{2s} \\ \vdots & & & \\ \mathbf{P}_{s1} & \mathbf{P}_{s2} & \dots & \mathbf{P}_{ss} \end{bmatrix} \tag{3.7.1}$$

where s is the total number of observation symbols in the process. In case of error source modeling, s takes only two values, either 0 or 1. The input sequence is in the form of a runlength vector and hence the length of the input sequence is greatly reduced, especially when we have sequences involving long stretches of identical symbols. Refer to section 2.11.2 for details about the runlength vector. The dimensions of the individual sub-matrices in the diagonal depend upon the number of all the states that can generate that particular observation symbol. The dimension of off-diagonal sub-matrices depends upon the number of states that generate the current observation symbol, and the number of states that can generate the next observation symbol. Define the fast forward variable $\boldsymbol{\alpha}(v_1^n, \tau)$ as

$$\boldsymbol{\alpha}(v_1^n, \tau) = \boldsymbol{\pi} \prod_{i=1}^{n_{t-1}} \mathbf{P}^{m_i}(X_i, \tau) \tag{3.7.2}$$

where $\boldsymbol{\pi}$ is the initial state probability vector,

X_i is the observation symbol, 0 or 1 in case of error sequences,

m is the number of times a particular symbol is observed.

The fast forward variable $\boldsymbol{\alpha}(v_1^n, \tau)$ is a row vector and is obtained in the vector form during the implementation of the algorithm. Define

$$\boldsymbol{\alpha}_0(v_1^t, \tau) = \pi_{X_1} \prod_{i=1}^{t-1} [P_{X_i X_i}^{m_i - 1} P_{X_i X_{i+1}}] \tag{3.7.3}$$

and

$$\boldsymbol{\alpha}(v_1^t, \tau) = \pi_{X_1} \prod_{i=1}^{t-1} [P_{X_i X_i}^{m_i - 1} P_{X_i X_{i+1}}] P_{X_i X_i}^{m_i - 1} \tag{3.7.4}$$

and hence develop the fast forward variable

$$\begin{aligned}
&\boldsymbol{\alpha}_0(v_1^0, \tau) = \boldsymbol{\pi}_{X_1} \\
&\boldsymbol{\alpha}(v_1^t, \tau) = \boldsymbol{\alpha}_0(v_1^t, \tau)\mathbf{P}_{X_i X_i}^{m-1} \\
&\boldsymbol{\alpha}_0(v_1^{t+1}, \tau) = \boldsymbol{\alpha}(v_1^t, \tau)\mathbf{P}_{X_i X_{i+1}}
\end{aligned} \tag{3.7.5}$$

Note that all these variables are row vectors. The dimension of these variables at a particular time depends upon the observation symbols observed at that time and the number of states that can generate that observation symbol. For example, if at time t a symbol 1 is observed and there are k states that can output a symbol 1, then the dimension of the fast forward variable is k. More precisely, the fast forward variable is of the order $1 \times k$.

The backward variable $\boldsymbol{\beta}(v_n^N, \tau)$ is shown as

$$\boldsymbol{\beta}(v_n^N, \tau) = \prod_{i-n_t}^{N} \mathbf{P}^{m_i}(X_i, \tau)\mathbf{1} \tag{3.7.6}$$

Note that this is a column vector. Define

$$\boldsymbol{\beta}_0(v_n^N, \tau) = \prod_{i=t}^{T} [\mathbf{P}_{X_{i-1} X_i} \mathbf{P}_{X_i X_i}^{m_i - 1}]\mathbf{1} \tag{3.7.7}$$

$$\boldsymbol{\beta}(v_t^N, \tau) = \mathbf{P}_{X_t X_t}^{m_t - 1} \prod_{i=t+1}^{T} [\mathbf{P}_{X_{i-1} X_i} \mathbf{P}_{X_i X_i}^{m_i - 1}]\mathbf{1} \tag{3.7.8}$$

Hence we derive the backward algorithm

$$\begin{aligned}
&\boldsymbol{\beta}_0(v_{N+1}^N, \tau) = \mathbf{1} \\
&\boldsymbol{\beta}(v_t^N, \tau) = \mathrm{Pr}_{X_t X_t}^{m_t - 1} \boldsymbol{\beta}_0(v_{t+1}^N, \tau) \\
&\boldsymbol{\beta}_0(v_t^N, \tau) = \mathrm{Pr}_{X_t X_{t+1}} \boldsymbol{\beta}_0(v_{t+1}^N, \tau)
\end{aligned} \tag{3.7.9}$$

Note that all of the above variables are in the form of a column vector. The dimension of these variables at a particular time depends upon the observation symbols observed at that time and the number of states that can generate that observation symbol. For example, if at time t a 1 is observed and there are k states that can output a symbol 1,

then the dimension of the fast backward variable is k and the order is $k \times 1$. More detail about this algorithm can be obtained from [6] and [12].

In order to estimate the model Λ, define

$$\Gamma_t(i, j) = \Pr(q_t = i, q_{t+1} = j \mid O, \Lambda) \tag{3.7.10}$$

This can be written in terms of the forward and backward variables $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as

$$\Gamma_{\boldsymbol{\mu}_c \boldsymbol{\mu}_{c+1}, c} = \frac{\alpha_{c|\boldsymbol{\mu}_c} \Lambda_{\boldsymbol{\mu}_c \boldsymbol{\mu}_{c+1}}^{m(\boldsymbol{\mu}_{c+1})-1} \beta_{c+1|\boldsymbol{\mu}_{c+1}}}{\Pr(O \mid \Lambda)} \tag{3.7.11}$$

$\Gamma_{\boldsymbol{\varepsilon}\boldsymbol{\mu}} = [0]$, unless $\boldsymbol{\varepsilon} = \boldsymbol{\mu}_c$ and $\boldsymbol{\mu} = \boldsymbol{\mu}_{c+1}$

Also

$$\Gamma_{\boldsymbol{\mu}_c \boldsymbol{\mu}_c, c} = \frac{(m(\boldsymbol{\mu}_c) - 1) \alpha_{c|\boldsymbol{\mu}_c} \Lambda_{\boldsymbol{\mu}_c \boldsymbol{\mu}_c}^{m(\boldsymbol{\mu}_c)-1} \beta_{c+1|\boldsymbol{\mu}_{c+1}}}{\Pr(O \mid \Lambda)} \tag{3.7.12}$$

where [0] denotes the matrix containing all zeros. In simulations, the probability $P(O \mid \Lambda)$ is not necessarily required.

The expected number of transitions from i to j is $\sum_{t=1}^{C-1} \Gamma_t(i, j)$, and this can be used to estimate the state transition matrix $\mathbf{P}$. Note that the elements of transition matrix can be estimated in the form of blocks, and hence the simulation runtime can be significantly reduced.

3.8 Forward-Only Semi-Markov Models

The forward-only semi-Markov model is obtained from the forward-only fast algorithm.

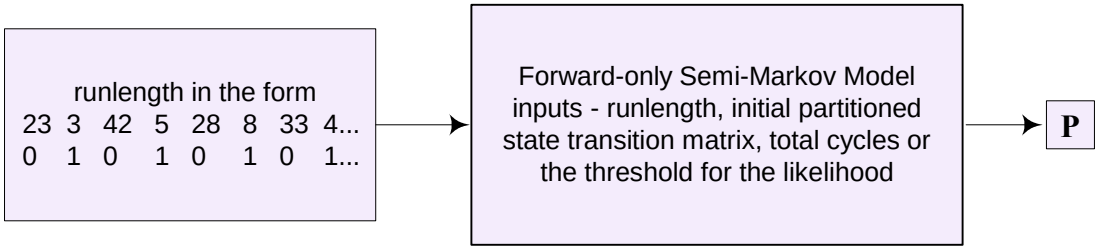

Figure 3.8.1: Forward-only semi-Markov model flow graph

We use modifications in the algorithm given in [6 and 37] to obtain this algorithm. This algorithm does not need the fast backward variable and hence it is capable of estimating the parameters of Markov models *on the fly*. As this model uses the runlength vector as input, hence it is fast as compared to the models discussed earlier. These properties allow it to be programmed in a DSP chip for real time applications. Consider the state transition matrix that was discussed for forward-backward semi-Markov models. The matrix has the form

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} & \dots & \mathbf{P}_{1s} \\ \mathbf{P}_{21} & \mathbf{P}_{22} & \dots & \mathbf{P}_{2s} \\ \vdots & & & \\ \mathbf{P}_{s1} & \mathbf{P}_{s2} & \dots & \mathbf{P}_{ss} \end{bmatrix} \tag{3.8.1}$$

where s is the total number of observation symbols in the process.

Define the fast forward variable $\boldsymbol{\alpha}(v_1^n, \tau)$ as

$$\boldsymbol{\alpha}(v_1^n, \tau) = \boldsymbol{\pi} \prod_{i=1}^{n_{t-1}} \mathbf{P}^{m_i}(X_i, \tau) \tag{3.8.2}$$

where $\boldsymbol{\pi}$ is the initial state probability vector,

X_i is the observation symbol, 0 or 1 in case of error sequences,

m is the number of times a particular symbol is observed.

The fast forward variable $\boldsymbol{\alpha}(v_1^n, \tau)$ is a row vector and is obtained in the vector form during the implementation. Define

$$\boldsymbol{\alpha}_0(v_1^t,\tau)=\pi_{X_1}\prod_{i=1}^{t-1}[\Pr_{X_iX_i}^{m_i-1}\Pr_{X_iX_{i+1}}] \tag{3.8.3}$$

and

$$\boldsymbol{\alpha}(v_1^t,\tau)=\pi_{X_1}\prod_{i=1}^{t-1}[\Pr_{X_iX_i}^{m_i-1}\Pr_{X_iX_{i+1}}]\Pr_{X_iX_i}^{m_i-1} \tag{3.8.4}$$

and hence develop the fast forward variable

$$\begin{aligned}&\boldsymbol{\alpha}_0(v_1^0,\tau)=\boldsymbol{\pi}_{X_1}\\&\boldsymbol{\alpha}(v_1^t,\tau)=\boldsymbol{\alpha}_0(v_1^t,\tau)\mathbf{P}_{X_iX_i}^{m-1}\\&\boldsymbol{\alpha}_0(v_1^{t+1},\tau)=\boldsymbol{\alpha}(v_1^t,\tau)\mathbf{P}_{X_iX_{i+1}}\end{aligned} \tag{3.8.5}$$

Note that all of these variables are row vectors. The dimension of these variables at a particular time is the number of states in the model.

We define $\mathbf{s}_{ij}(\boldsymbol{x}_1^{t+1})$ as $\mathbf{s}_{ij}(\boldsymbol{x}_1^{t+1})=\mathbf{s}_{ij}(\boldsymbol{x}_1^{t})\mathbf{P}^{m_{t+1}}+\boldsymbol{\alpha}(\boldsymbol{x}_1^{t_k})\mathbf{V}$ (3.8.6)

where the matrix $\mathbf{V}$ is calculated using the following

$$\begin{aligned}&b=\text{rem}(m,2) && \mathbf{V}=b\mathbf{W}\\&\mathbf{R}=\mathbf{P} \quad \mathbf{Q}=\mathbf{P}^b && \mathbf{G}=\mathbf{W}\\&\text{while } m>0\\&\quad \mathbf{G}=\mathbf{R}\,\mathbf{G}+\mathbf{G}\,\mathbf{R} && \mathbf{R}=\mathbf{R}^2\\&\quad m=(m-b)/2 && b=\text{rem}(m,2)\\&\quad \mathbf{V}=\mathbf{R}^b\,\mathbf{V}+b\,\mathbf{G}\,\mathbf{Q} && \mathbf{Q}=\mathbf{Q}\,\mathbf{R}^b\\&\text{end}\end{aligned} \tag{3.8.7}$$

This allows us to use fast matrix exponentiation which is very useful in reducing the runtime of the algorithm because the value of m could be very large, as the lengths between two consecutive bursts might be very long. The vector $\mathbf{s}$ is added to get the state transition matrix

$$\mathbf{P}(i,j)=\sum_{i=1}^{N}\mathbf{s}_{ij}(x_1^{t+1}) \qquad t=1,2,...,T \tag{3.8.8}$$

Using the above procedure, the state transition matrix can be estimated. This a is forward-only semi-Markov model, as it does not require the backward path and it also uses the

runlength vector for faster computations. This model might give us sub-optimal results as we are trading off the speed for computational complexity. We will be elaborate more clearly when we discuss its simulation results in Chapter 5. Further modifications are underway to improve this algorithm for both accuracy and computational efficiency. This algorithm can replace most of the existing algorithms that are available, and due to its very high computational efficiency it can be a useful tool for the fast estimation of Markov model parameters.

3.9 Initial Model Parameter Estimate

When we normally try to build an HMM we have nothing but streams of sequences. Sometimes we have parameters from old tested models, but this is not normally the case. To set the initial model parameters, we need to be careful, as one might slip into divergence with bad initialization. The problem with discrete observations HMM is less effective as we can initialize the model parameters with random values, with the constraints mentioned in section 2.3. The initialization problem is more serious when we have continuous hidden Markov models (CHMM) and the parameters should be selected to get rid of the divergence fate.

3.10 Number of States of a Model

There is no straightforward answer to the question of how many numbers of states should be specified to optimally define a model. Sometimes previous experience about the problem is necessary or one has to suggest a different number of states and then select the one that gives the best results. Increasing the number of states sometimes gives sub-optimal results. Hence one should not expect more accurate results just by increasing the number of the states of the model. References that discuss this issue are [12, 52, 56 and 60].

3.11 Summary

In this chapter Markov models with their mathematical descriptions are discussed. These models can easily be implemented using C or MATLAB. In Chapter 5 simulation results for these models are shown.

Chapter

4

Simulation: Architecture and Methodology

This chapter discusses the simulation architecture and methodology for modeling third generation wireless channels using Markov models. We give details about the Wideband Code Division Multiple Access (W-CDMA) simulator and then discuss the Vector ImPulsE Response (VIPER) measurement system, which is a tool used for channel modeling. Markov modeling is discussed in Section 4.2.

Figure 4.1 shows the simulation architecture of the waveform level W-CDMA simulator. The simulator is designed for both accuracy (relative to the W-CDMA standard) and speed of execution. The simulator can support any specular multipath environment, any number of interferers, and Doppler frequencies up to 400 Hz. For simplicity of design and integration, the carrier frequency is fixed at 1970 MHz. The execution speed of this *frame based, waveform simulator*, depends on the complexity of the multipath channel and the number of interferers. For an additive white Gaussian noise (AWGN) channel with no interferers, the simulator can process one frame in 1.2 seconds on a Pentium III, 900 MHz computer. For a four ray multipath channel with five interfering signals, the simulator takes 6 seconds to process one frame.

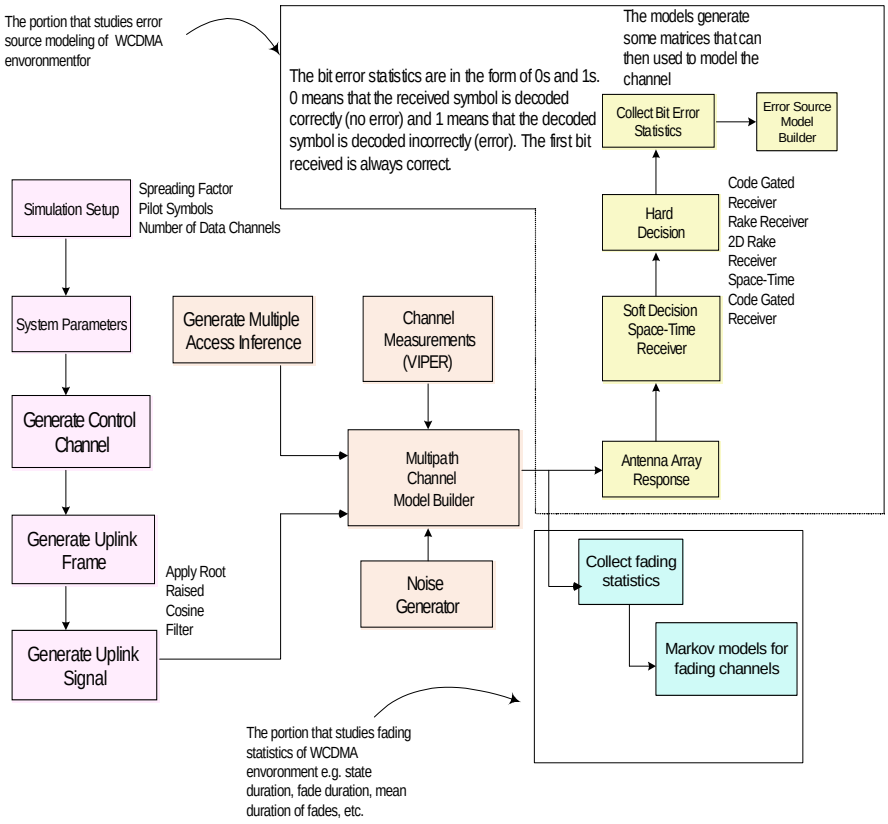

Figure 4.1: Simulation architecture of W-CDMA simulator with VIPER input

The power delay profiles from VIPER are used as input to the simulator, as shown in Figure 4.1. This figure illustrates different portions of the simulator. Details about VIPER and the error source model builder are discussed further in later sections along with a discussion about fade duration modeling in wireless channels.

As an example, we parameterize the W-CDMA uplink simulator with 3 interferers, an E_b / N_o of 3dB and an assumed vehicle speed of 50 kilometer per hour. The power delay profiles from VIPER are taken as the input, and the threshold of –20 dB is set for the noise threshold. Signals below this threshold are considered noise.

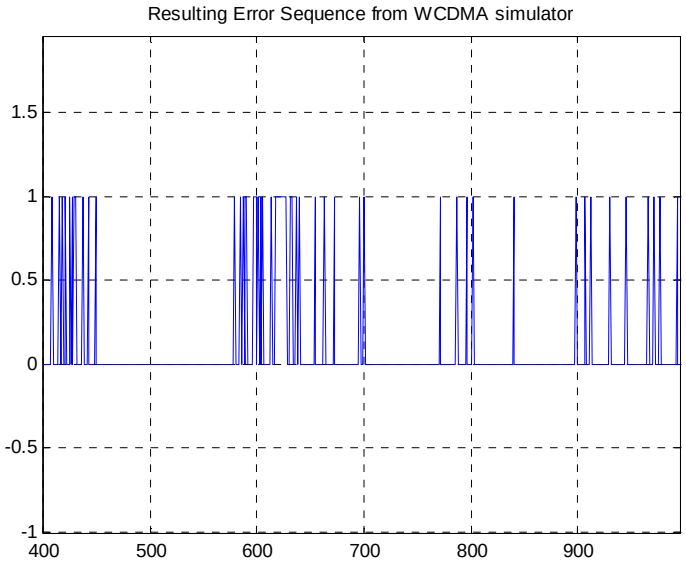

Figure 4.2: Error sequence obtained from W-CDMA simulator with VIPER input

The BER as determined by the W-CDMA simulator is found to be 0.1264. The error pattern is used as the input to the Markov model and this Markov model is then used to generate statistically similar error sequences.

4.1 Vector ImPulsE Response Measurement System (VIPER)

This section describes the design and development of a wideband, multi-channel, real-time, software-defined measurement receiver [73]. A *measurement receiver* is a radio receiver whose purpose is to measure the characteristics of received signals and the channels through which the signals propagate. A *real-time measurement receiver* presents signal and channel results as measurements are performed and at a rate sufficient to characterize the time-varying nature of the signals and channels, specific to the characterization parameters used.

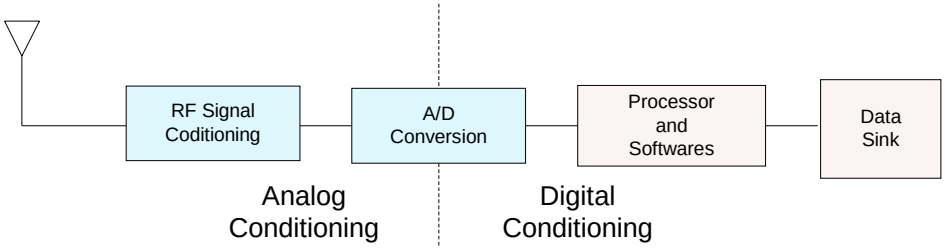

Figure 4.1.1: Wideband, multi-channel, real-time, software-defined measurement receiver

The measurement receiver has been successfully built, demonstrated, and used in the field for RF channel measurements. The measurement receiver has also served as a testbed for smart antenna algorithms. This receiver can be programmed using MATLAB or C++. Since MATLAB is a widely used tool for simulating communication systems, the MATLAB interface capability for the measurement receiver provides a straightforward way to turn simulated processing algorithms into functional software radio modules to process actual received radio signals. The modularity of the software facilitates managing the code for expansion to future applications.

Figure 4.1.2: Measurement system setup for the receiver

Virginia Tech has completed wideband measurements to characterize radio channels at a receiver height of 30 meters. The wideband, multi-channel measurement system is mounted on the roof of Whittemore Hall, a six-story academic building on the Virginia

Tech campus. The transmitter antenna is mounted on the roof of a vehicle which is driven through the parking lots and streets adjacent to the building on which the receiver is located. Figure 4.1.2 shows the measurement system location on the roof and the orientation of the antenna array relative to the surroundings.

The measurement system is positioned on the roof of Whittemore Hall near the corner of the building. The antennas are mounted on a stand approximately six feet above roof level. The antenna array used at the receiver is a four-element linear array, using monopole antenna elements with half-wavelength spacing. The antenna array is mounted so that the ground plane is above the antenna elements. In this configuration the antennas receive signals from below the horizontal plane (where the transmitter would be located during driving). Table 4.1.1 summarizes the configuration details.

Transmit power	+26 dBm
Transmitted signal	80 Mcps PN sequence, 1023 chips, register taps (3,10)
Transmit Frequency	2050 MHz (center)
Transmitter antenna	Dipole, vertically-polarized
Transmitter antenna height	1.5 m AGL
Receiver antennas	Four-element monopole array, half-wavelength spacing
Receiver antenna array height	Approximately 30 m AGL
Receiver location	Whittemore Hall, roof
Drive test areas (transmitter driven)	1) Parking lots north of Whittemore (LOS) 2) Parking lots behind Durham (NLOS) 3) Suburban neighborhood north of Whittemore(LOS/NLOS)

Table 4.1.1: VIPER measurements details

The automobile with the transmitter is driven at slow speeds (less than 5 MPH) while the receiver logs signal data. GPS position data of the transmitter is collected for a majority of the signal snapshots.

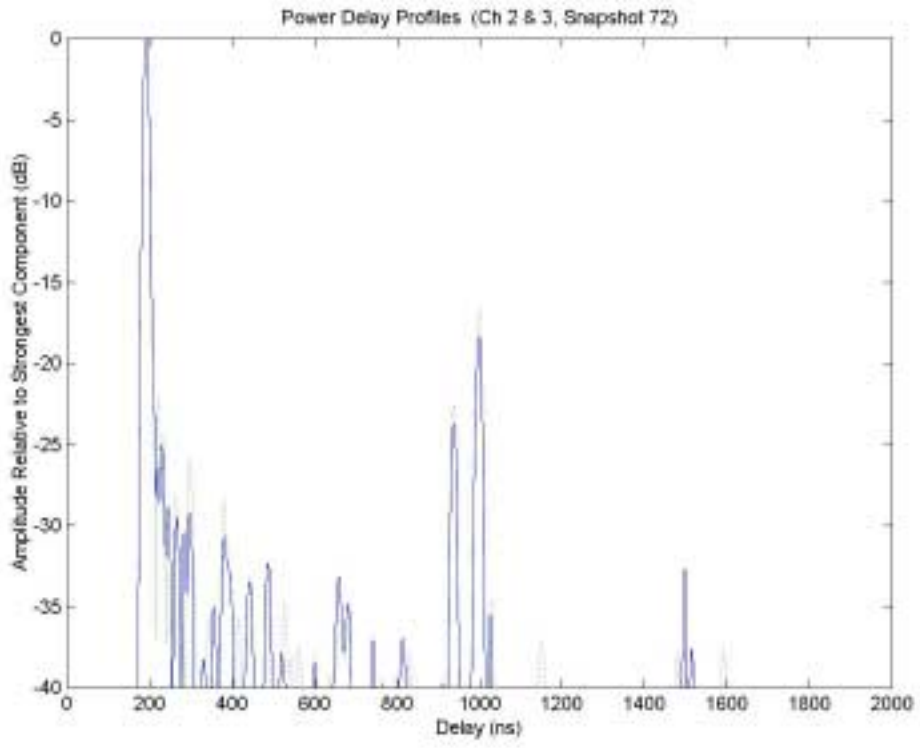

Figure 4.1.3: Sample PDP recorded for antenna two (solid) and three (light dotted).

The GPS position of the receiver array is also noted. Raw signal data is recorded to disk, and the measurements are post-processed. Figure 4.1.2 shows sample power delay profiles (PDPs) produced from data recorded simultaneously at channels 2 and 3 of the receiver. The profiles show evidence of uncorrelated fading of the multipath components.

Table 4.1.2 describes multipath delay statistics derived from the logged measurements. RMS delay spread statistics are computed using the measurements taken between the roof receiver site and the parking lot north of Whittemore Hall. These measurements will assist with future experiments performed in the vicinity of Whittemore Hall.

Parameter	*Channel 1*	*Channel 2*	*Channel 3*	*Channel 4*
Mean RMS Delay Spread	116 ns	184 ns	218 ns	207 ns
Standard Deviation RMS Delay Spread	105 ns	103 ns	120 ns	98 ns
Maximum RMS Delay Spread	373 ns	460 ns	450 ns	404 ns

Table 4.1.2: VIPER RMS delay spread results

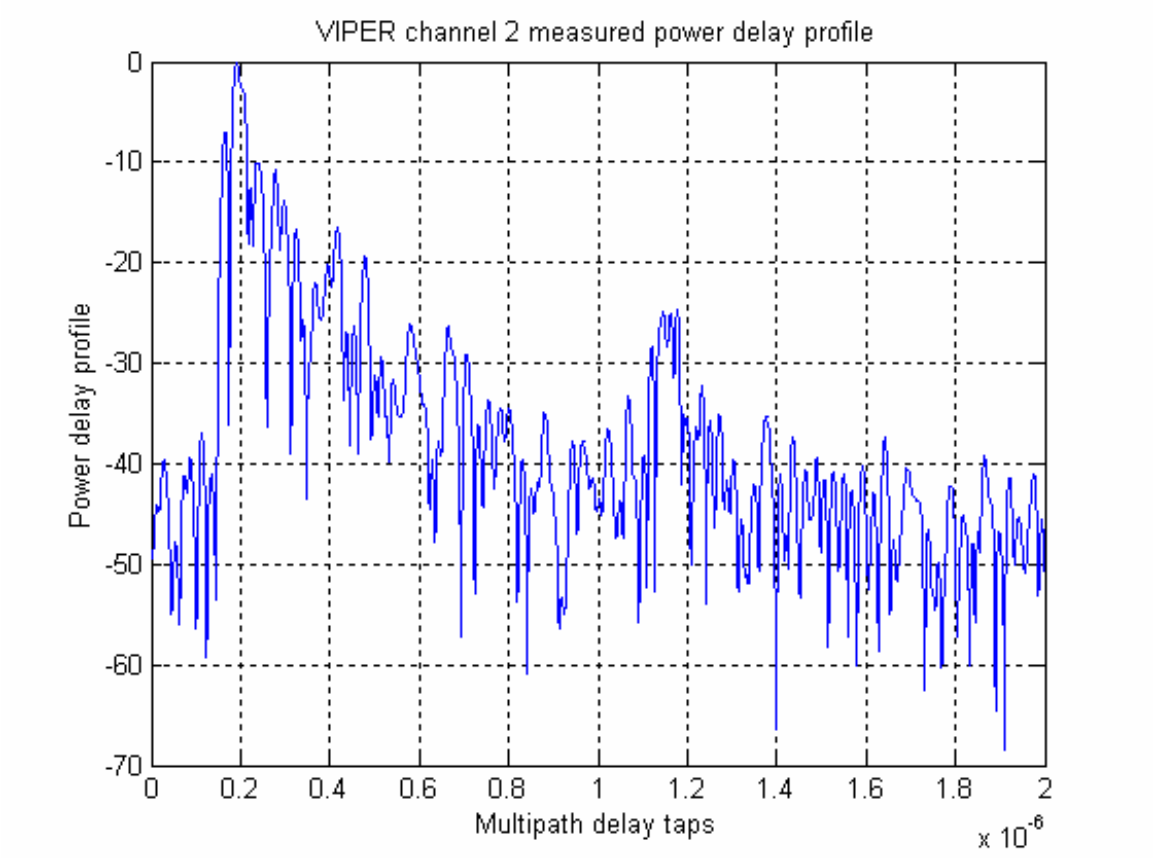

Figure 4.1.4: VIPER Channel 2 power delay profile

Table 4.1.2 shows a wide spread in RMS delay spread results. The largest RMS delay spread is computed to be 460 ns. However, for some profiles containing only very weak multipath components, the RMS delay spread is below 20 ns.

Figure 4.1.4 shows a typical power delay profile from VIPER. These power delay profiles are set to a certain threshold (normally –20 dB). Decreasing the threshold level increases the length of the power delay profile vector.

4.2 Markov Modeling

Error statistics in the form of zeros and ones (a binary one denotes an error) are obtained from the waveform level simulator (W-CDMA simulator). These error sequences are then used for modeling error sources using different Markov models, i.e., hidden Markov models and semi-hidden Markov models. The tradeoffs in using these different models are discussed in a later section of this thesis.

Figure 4.2.1 shows different error source models that will be used in the simulations to follow. Once we have a reliable model from the channel of interest, we can discard the W-CDMA simulator and generate the similar bit error statistics from these models. The error sequences generated by these models are not point to point identical but are statistically equivalent. For very complex environments in which many interferers and several multipath are present, the runtime of the waveform level simulation becomes very long. However with these models we can generate similar results with greatly reduced runtime.

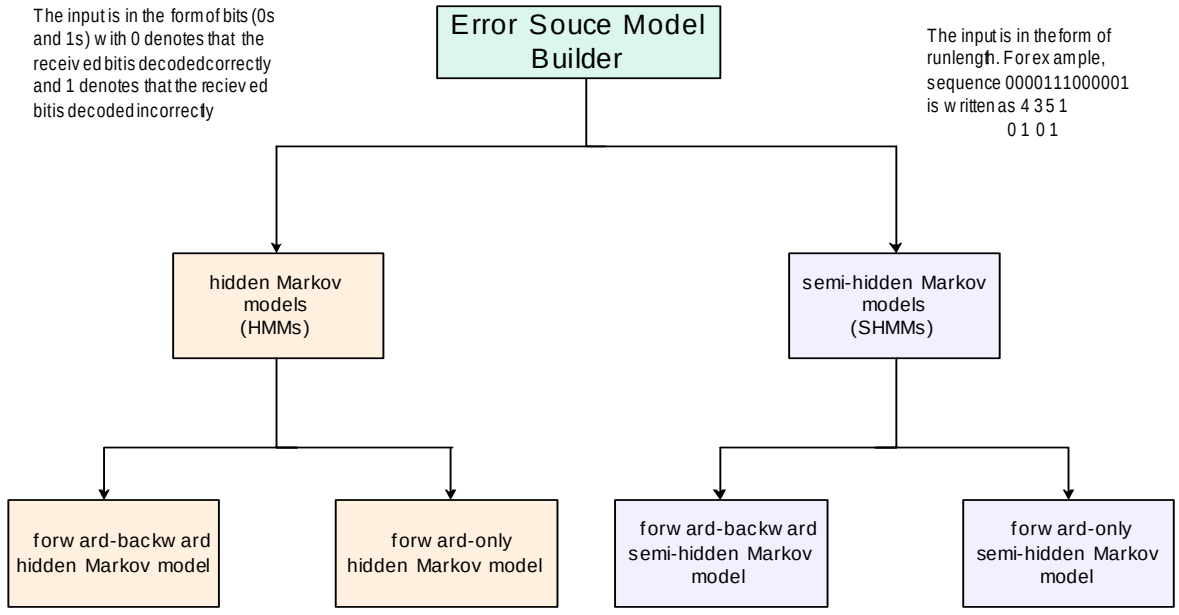

Figure 4.2.1: Error source models

Several different models for generating error sources in the W-CDMA environment will be used. These models can be broadly classified on the basis of the input sequences that are used to form the state transition matrix.

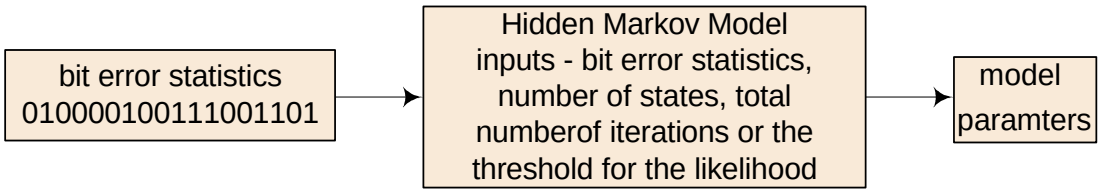

Figure 4.2.2: Hidden Markov Model flow graph

Hidden Markov models use the error vector in a bit-by-bit basis as input to the model-building algorithm. Modeling development can have very long runtime if the sequences have long stretches of identical symbols. In our case, the symbols are generally zeroes or ones, with a one denoting a single error. Figures 4.2.2 and 4.2.3 show the simulation

architecture of HMMs and semi-HMMs respectively. Semi-HMMs use the runlength vector, as was discussed in Chapter 2. Using runlength vectors sometimes might give us sub-optimal results, but these models are tested and were proved sufficiently reliable for W-CDMA environments. Other statistics, e.g., autocorrelation and the number of error bursts were evaluated, and we observed that these models can provide sequences that match closely with the error sequences obtained from the W-CDMA simulators.

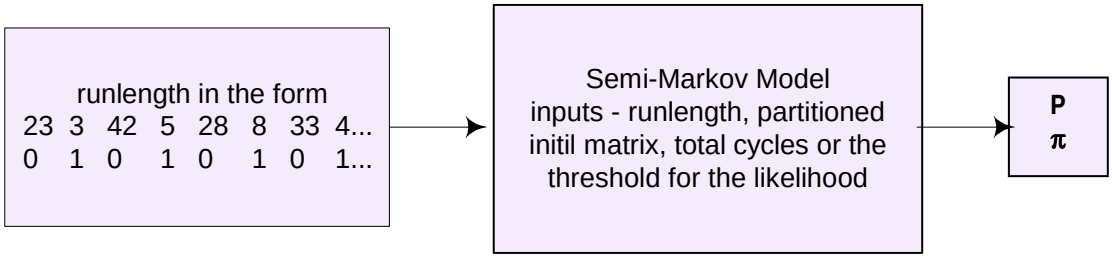

Figure 4.2.3: Semi-hidden Markov model flow graph

Descriptions about these models and the algorithms that produced these models were discussed in the Chapter 3.

According to [32], if fading can be modeled by an HMM, bit errors and block errors in data communications over fading channels can also be modeled by HMM. We will validate these properties in our case with W-CDMA fading statistics.

We use the BWA for PH-distribution to model fade durations in the wireless channels. Fading envelopes are quantized to several levels whose selection is application dependent. It is usually possible to select the quantization levels in such a way that the level change is a Markov chain. In this case we have a semi-Markov process, and in order to model this process with an HMM, we need to approximate the state durations with the PH distributions and then construct the HMM.

For wireless channel simulations, the underlying Markov models can have a varying number of states depending upon the complexity of the channels (or the complexity of error patterns, as in case of error source modeling). For the case of error source modeling, the probability of error itself is not sufficient for determining the reliability of the model because there can be more than one error sequence having exactly the same BERs but

being very different in nature. This raises the question of how many error bursts of specific lengths will be there in the error sequence.

Example 4.2.1

Consider an error sequence as shown in the Figure 4.2.1. The sequence shown in Figure 4.2.1 has a bit error rate of 0.1 since there are 10 errors bits in the sequence of 100 bits.

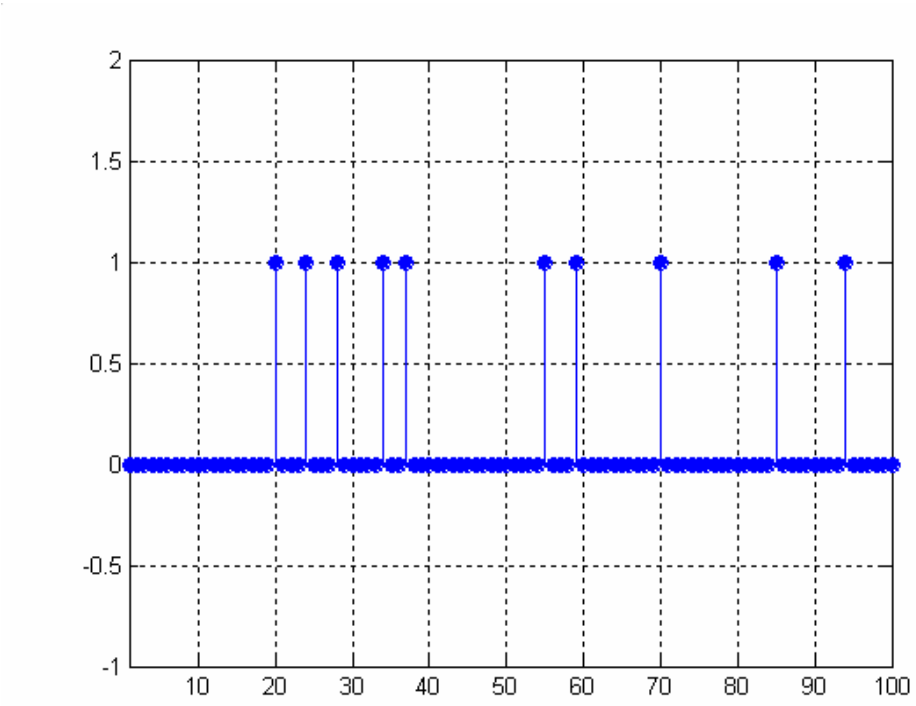

Figure 4.2.1.1: Error sequence 1

Figure 4.2.2 shows the error sequence having the same BER of 0.1, but here errors are occurring in the form of bursts. Note that the nature of this error sequence is very different from the sequence shown in Figure 4.2.1.

The length between two consecutive bursts should also be considered and should be modeled correctly. In Chapter 5 we will attempt to analyze and answer these questions using computer simulations. Another important issue is the convergence and robustness of models for different training sequences.

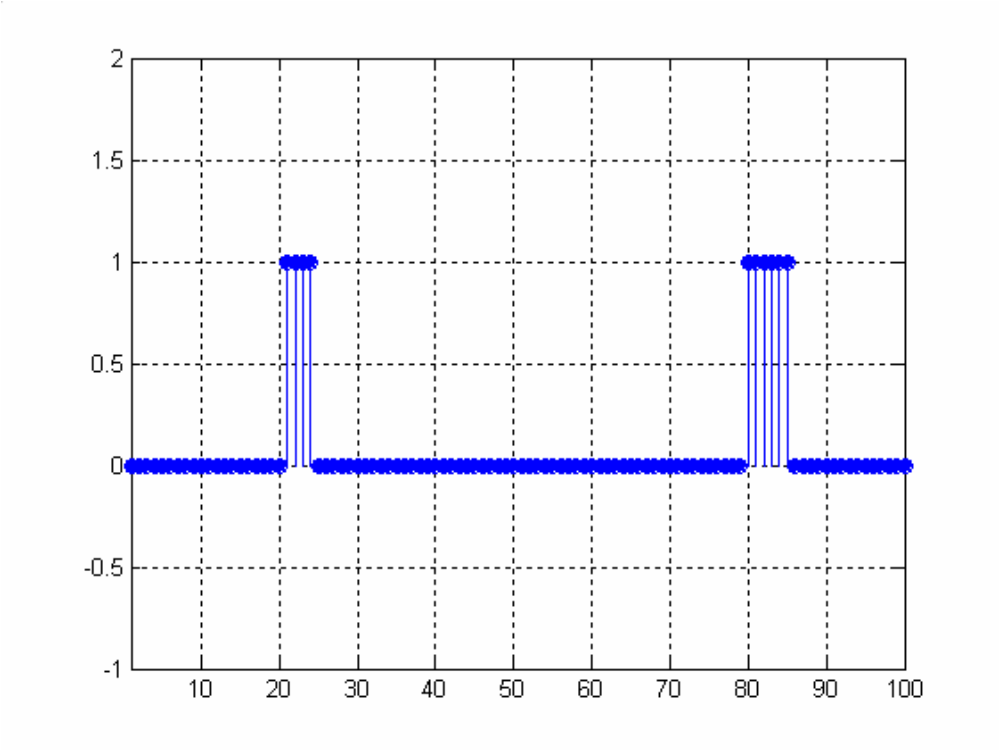

Figure 4.2.1.2: Error sequence 2

4.3 Summary

The architecture of the W-CDMA simulator was discussed in this chapter. The wideband real world vector channel receiver (VIPER) was also discussed in a later section. The purpose of using VIPER is to provide real world channel measurements to the W-CDMA simulator for getting more practical and accurate results. VIPER provides real-world channel power delay profile measurements as input to the simulator. Different Markov modeling techniques were discussed in greater details. Once statistical results from the W-CDMA simulator are obtained, we use different Markov modeling techniques to model these statistical results with greatly reduced computer memory and simulation runtime.

Chapter

5

Simulation Results

In this chapter, Markov model simulation results obtained primarily for third generation wireless channels are discussed. The models that were discussed in Chapter 4 are tested for their closeness to the waveform level simulation results. Simulation results for error source modeling and fade duration distributions are discussed, with emphasis on W-CDMA channels.

5.1 Fade Duration Distribution Approximations

PH distributions approximations are used to model the fade durations in wireless channels. Results of both flat fading and frequency selective channels are discussed here. We take the fading envelope and quantize it to different levels. The number of levels defines the number of states used in the model.

First, a waveform level simulation was performed using Clarke's model [45] of the Rayleigh fading channel with Doppler parameter $f_D = 100\,\text{Hz}$, and a normalized fade rate $f_D T = 0.01$ [44]. We first validate and compare our results for the case of Rayleigh fading that is given in [6]. The parameter and quantization values are consistent with [6].

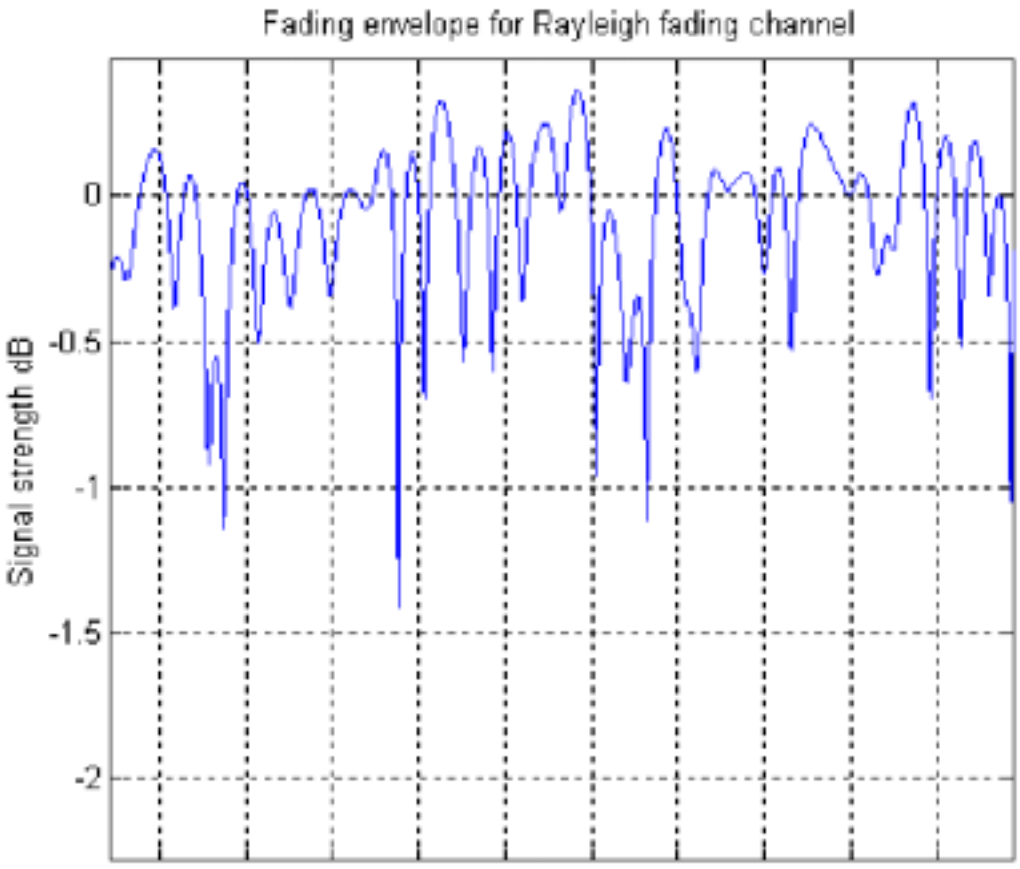

Figure 5.1.1: Rayleigh fading envelope

We simulated 1 million steps of the fading amplitude to estimate the state transition probability matrix where the states are represented by the quantization intervals (0, 0.0149), (0.0149, 0.0329), (0.0329, 0.0725), (0.0725, 0.1601), (0.1601, 0.3535), (0.3535, 0.7805), (0.7805, 1.7234) and $(1.7234, \infty)$. Figure 5.1.1 shows the small portion of the Rayleigh fading envelope that is used here in simulation.

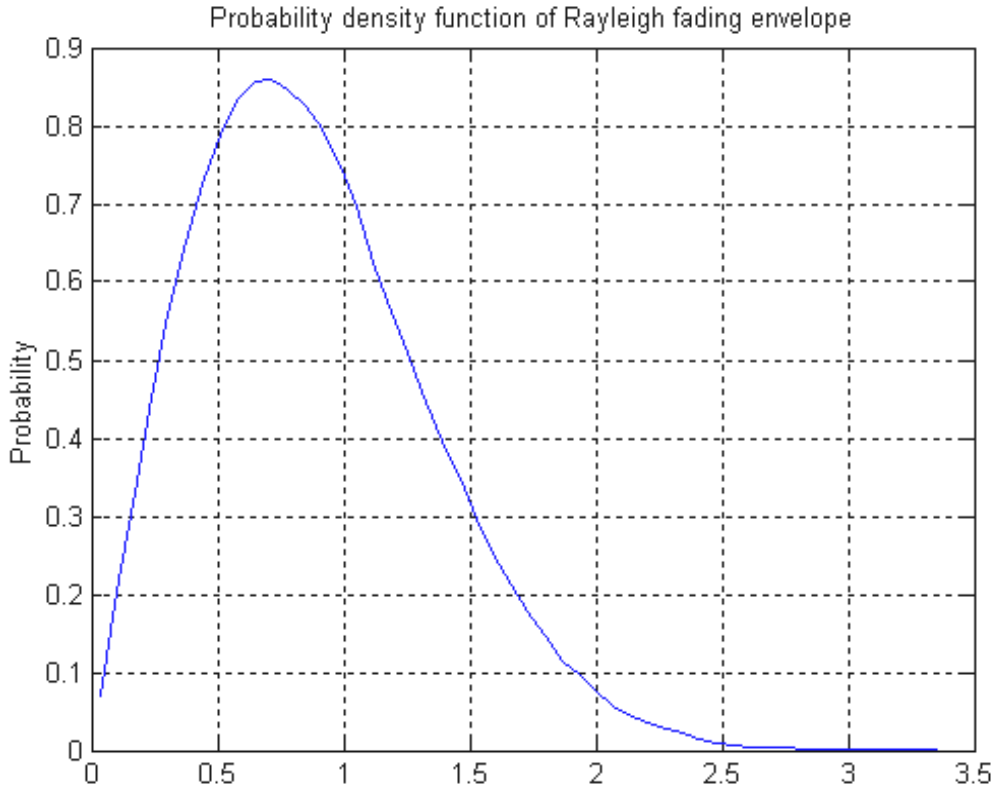

Figure 5.1.2: Probability density function of Rayleigh fading envelope

Figure 5.1.2 shows the probability density function of the Rayleigh fading envelope. The quantized levels of the fading envelope are shown in Figure 5.1.3. There are 8 quantization levels, and the sixth quantization level is selected for this simulation. The primary reason for selecting the sixth quantization level is that it contains the largest number of data samples and we expect more accurate results as the number of data samples increases. This sixth quantization level is then approximated using a 10-state PH distribution approximation model.

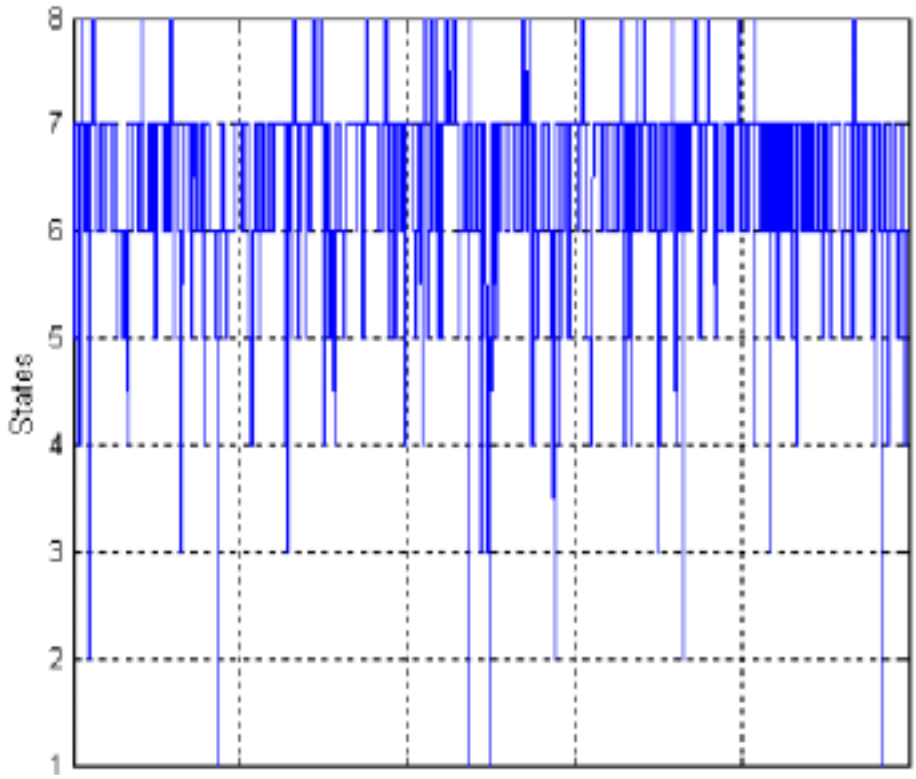

Figure 5.1.3: Fading envelope quantized to different fading levels

The quantized state distribution vector is given as input to the model that approximates the PH-distribution, i.e. the model discussed in section 3.4. The matrix that is obtained from the PH-distribution approximation algorithm is shown below. Note that this has the same form as defined in (3.4.2).

$$\mathbf{P} = \begin{bmatrix} 0.1460 & 0.0476 & 0.0294 & 0.0013 & 0.1955 & 0.0482 & 0.0793 & 0.3700 & 0.0821 & 0.0006 \\ 0.0275 & 0.1972 & 0.0033 & 0.2295 & 0.0001 & 0.0564 & 0.0910 & 0.0014 & 0.3935 & 0.0000 \\ 0.0095 & 0.1242 & 0.1889 & 0.0411 & 0.0510 & 0.0106 & 0.0387 & 0.0243 & 0.0111 & 0.5006 \\ 0.1111 & 0.1173 & 0.0001 & 0.1897 & 0.0027 & 0.1059 & 0.1124 & 0.0224 & 0.3378 & 0.0008 \\ 0.0446 & 0.0456 & 0.1243 & 0.0897 & 0.2281 & 0.0175 & 0.1069 & 0.0453 & 0.0100 & 0.2881 \\ 0.0934 & 0.0253 & 0.1031 & 0.0276 & 0.1120 & 0.1514 & 0.0885 & 0.3128 & 0.0811 & 0.0048 \\ 0.1093 & 0.1399 & 0.0266 & 0.1354 & 0.0709 & 0.1371 & 0.0912 & 0.2341 & 0.0540 & 0.0015 \\ 0.0011 & 0.0594 & 0.5455 & 0.0309 & 0.1182 & 0.0960 & 0.0242 & 0.1042 & 0.0108 & 0.0098 \\ 0.2230 & 0.0222 & 0.0058 & 0.0147 & 0.0080 & 0.2601 & 0.0606 & 0.2284 & 0.1772 & 0.0000 \\ 0.0010 & 0.7757 & 0.0001 & 0.2148 & 0.0004 & 0.0002 & 0.0007 & 0.0004 & 0.0068 & 0 \end{bmatrix}$$

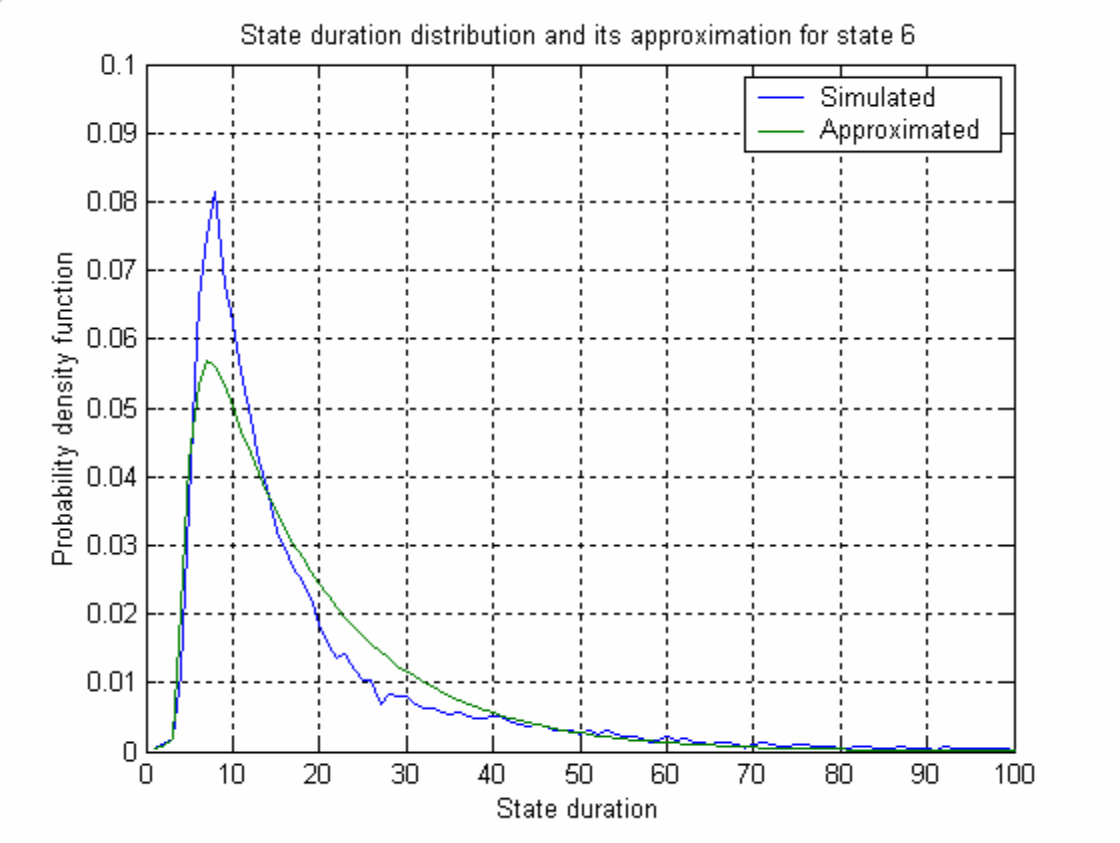

Figure 5.1.4: A typical state duration distribution and its approximation

The state duration distribution and its approximation for a typical Rayleigh fading channel are shown in Figure 5.1.4. We observe that this resembles a Gamma distribution

[61]. Now we apply the same procedure for the W-CDMA fading envelope for state 7 and obtain the result shown in Figure 5.1.5. The two curves match perfectly.

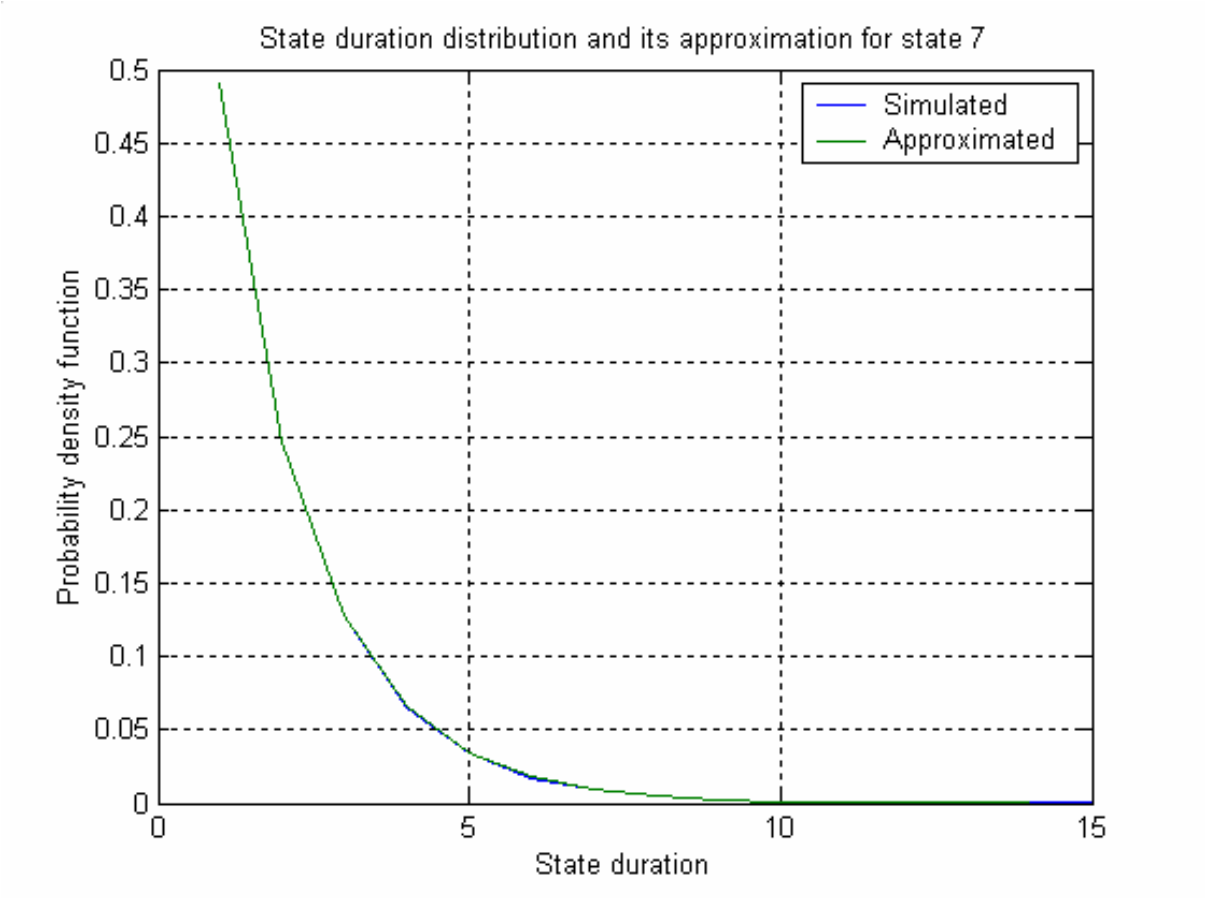

Figure 5.1.5: Approximation of state duration distribution for a typical W-CDMA channel

5.2 Error Source Simulations

In this section it will be shown that using Markov models, error sequences can be generated that are *statistically* similar to the sequences obtained from the W-CDMA simulator. An error sequence obtained from the W-CDMA simulator is used here as input to the Markov model algorithms. The process of error source simulation was discussed in detail in section 2.13.

An error sequence of 400,000 bits having probability of error of 0.0819 is obtained from the W-CDMA simulator.

5.2.1 Forward-Backward HMM Error Source Simulation

The error sequence that was obtained in last section is given as input to a 3-state forward-backward HMM. After 200 iterations, the estimated parameters of the HMM obtained are

$$\mathbf{P} = \begin{bmatrix} 0.9807 & 0.0027 & 0.0166 \\ 0.8407 & 0.0853 & 0.0740 \\ 0.3188 & 0.0124 & 0.6688 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.9253 & 0.2461 & 0.8248 \\ 0.0747 & 0.7539 & 0.1752 \end{bmatrix}$$

$$\boldsymbol{\pi} = [1.0000 \quad 0.0000 \quad 0.0000]$$

Let us define a matrix $\mathbf{\Phi}$ in such a way that

$$\mathbf{\Phi} = \mathbf{P}^n \tag{5.2.1}$$

where n is a large number.

Raising $\mathbf{P}$ to a high power makes the rows of the resulting matrix identical. This high number may be in the order of 10^{10}. We see that further increases in power do not change the elements of the resulting matrix. Keep in mind that the $\mathbf{P}$ matrix should strictly be a stochastic matrix and hence all of its rows must sum to 1. If the second row of the $\mathbf{B}$ matrix corresponds to the output symbol probability of generating an error, then the probability of error P_E is given as

$$\mathbf{\Phi} = \mathbf{P}^{100000000}$$

$$= \begin{bmatrix} 0.9482 & 0.0035 & 0.0483 \\ 0.9482 & 0.0035 & 0.0483 \\ 0.9482 & 0.0035 & 0.0483 \end{bmatrix}$$

$$\begin{aligned} P_E &= P(e,S_1) + P(e,S_2) + P(e,S_3) \\ &= P(e/S_1)P(S_1) + P(e/S_2)P(S_2) + P(e/S_3)P(S_3) \\ &= \mathbf{\Phi}(1,1)\mathbf{B}(2,1) + \mathbf{\Phi}(2,2)\mathbf{B}(2,2) + \mathbf{\Phi}(3,3)\mathbf{B}(2,3) \\ &= 0.0819 \end{aligned}$$

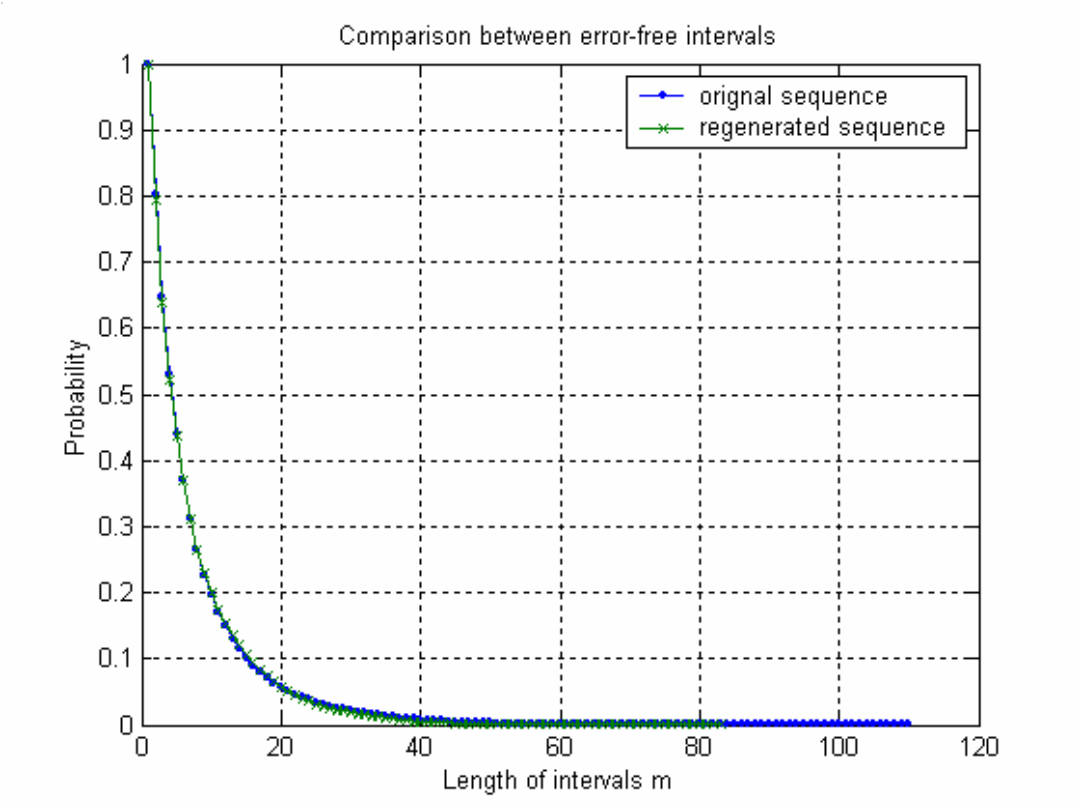

Figure 5.2.1.1: Forward-backward HMM error-free interval distribution comparison

Figure 5.2.1.1 shows the comparison between the error-free intervals of the original sequence and the sequence that was obtained using error source simulation. These plots match closely and therefore the result is assumed satisfactory.

5.2.2 Forward-Only HMM Error Source Simulation

Applying the same error sequence to the 3-state forward-only HMM, we get the following result:

$$\mathbf{P} = \begin{bmatrix} 0.92534 & 0.06923 & 0.00541 \\ 0.05606 & 0.83846 & 0.10547 \\ 0.21132 & 0.00277 & 0.78590 \end{bmatrix}$$

$$\mathbf{B} = \begin{bmatrix} 0.91931 & 0.91942 & 0.91947 \\ 0.08068 & 0.08057 & 0.08052 \end{bmatrix}$$

Note that due to the unavailability of the backward path, the initial state probability vector cannot be estimated. The error probability from this model is obtained as 0.08063, which matches closely with the results obtained using the forward-backward algorithm.

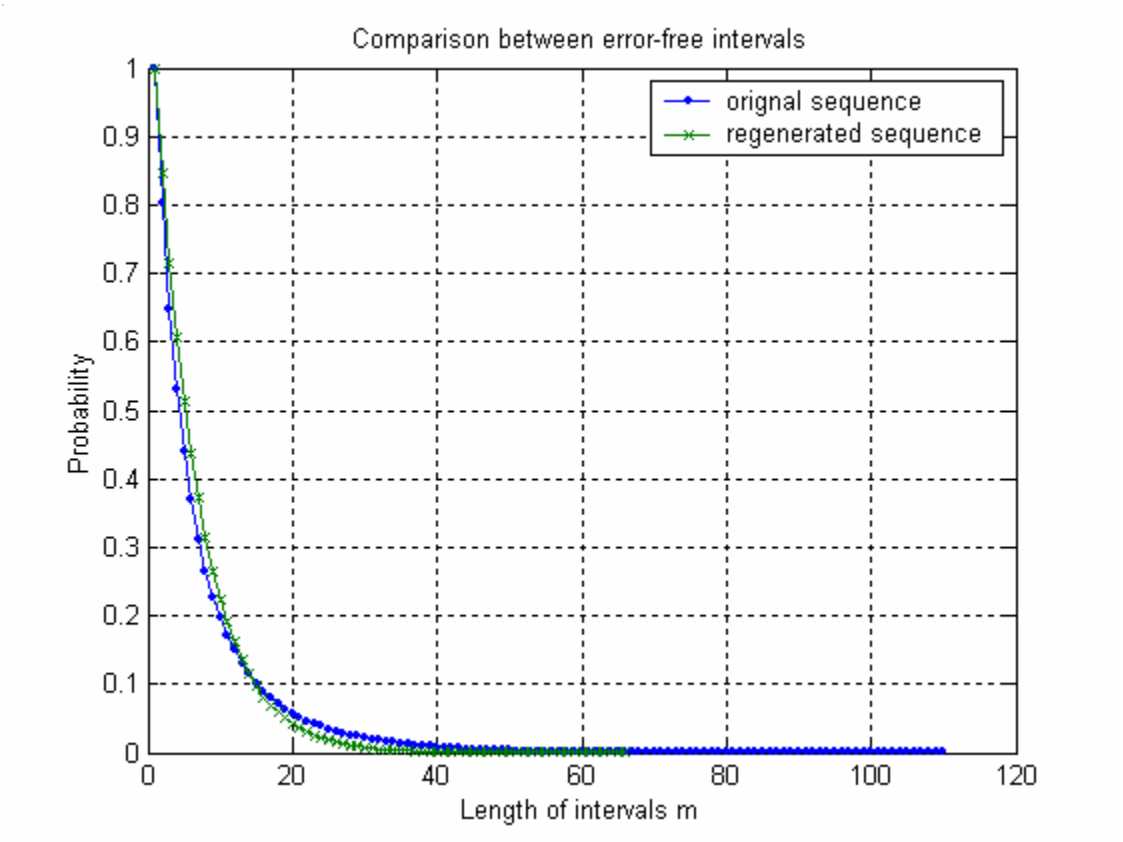

Figure 5.2.2.1: Forward-only HMM error-free interval distribution comparison

Figure 5.2.2.1 shows the error-free interval distributions between an error-source sequence and the sequence obtained from the forward-only HMM. From the plot it can be seen that these distributions show a close match.

5.2.3 Forward-Backward SHMM Error Source Simulation

Now consider the forward-backward semi-hidden Markov model with one good state and two bad states. So in this case

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

Note that the two zeroes show that transitions between the two bad states are not allowed. The state transition matrix obtained after 200 iterations is found as

$$\mathbf{P} = \begin{bmatrix} 0.91972 & 0.03829 & 0.04197 \\ 0.90653 & 0.09346 & 0 \\ 0.92218 & 0 & 0.07781 \end{bmatrix}$$

The $\boldsymbol{\pi}$ vector obtained from this model is given as

$$\boldsymbol{\pi} = \begin{bmatrix} 0.1846 & 0.0439 & 0.7714 \end{bmatrix}$$

The error probability from the model is found to be 0.08068, which is in good agreement with the two previously obtained results. Figure 5.2.3.1 shows the comparison of the error-free intervals for the case of the forward-backward semi-hidden Markov model. The error-free interval distribution comparison between the original sequence and the sequence obtained from the parameters of the forward-backward semi-HMM indicates that these two distributions match closely.

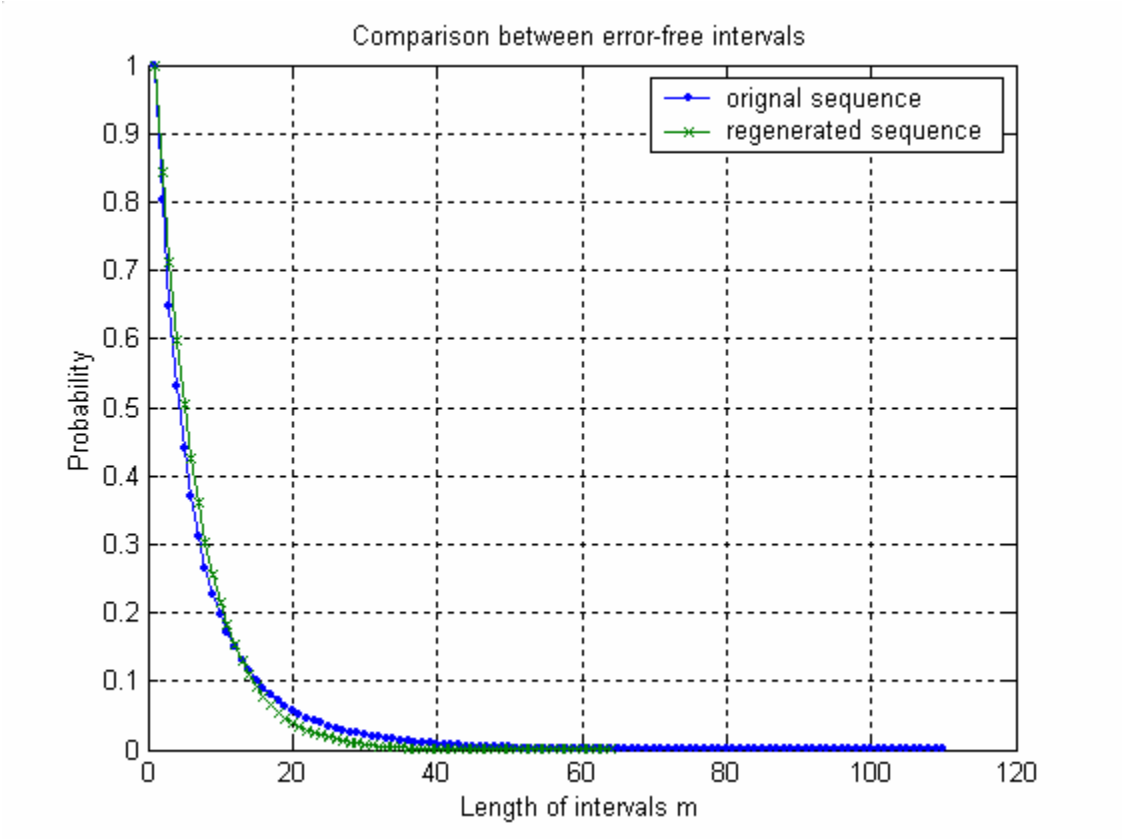

Figure 5.2.3.1: Forward-backward semi-HMM error-free interval distribution comparison

5.2.4 Forward-Only SHMM Error Source Simulation

Modeling the error sequence using the forward-only semi-hidden Markov model having one good and two bad states gives following result:

$$\mathbf{P} = \begin{bmatrix} 0.82011 & 0.08994 & 0.08994 \\ 0.99376 & 0.00623 & 0 \\ 0.99376 & 0 & 0.00623 \end{bmatrix}$$

The $\mathbf{B}$ matrix is once again assumed to have the form

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

The error probability of this model is obtained as 0.15234. This is not a very reliable model probably because we are putting too many constraints in the model. First the model follows the forward-only approach and secondly the model uses the runlength vector of the error sequence instead of the whole sequence.

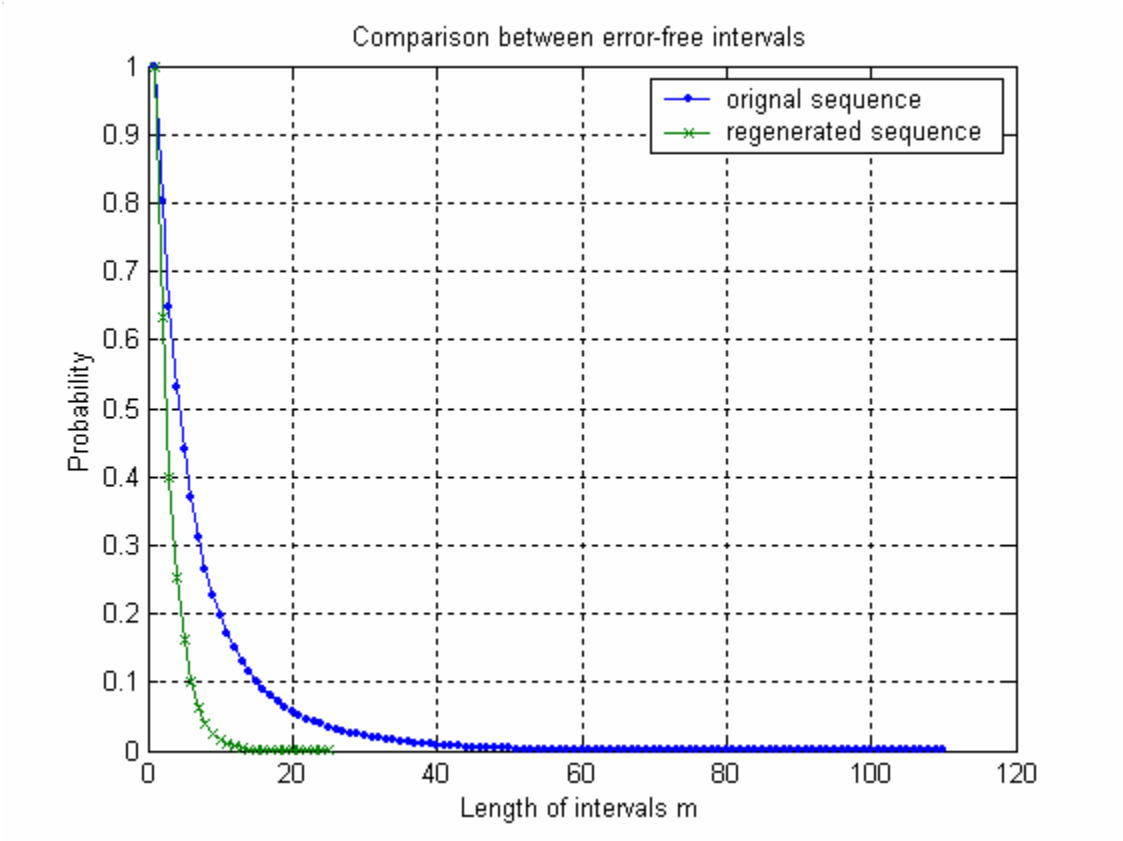

Figure 5.2.4.1: Forward-only semi-HMM error-free interval distribution comparison

Figure 5.2.4.1 shows that the distributions do not match properly. Improvements in this algorithm are under way and we expect to get better results in the future.

5.3 Hidden Markov Models BER Comparison

In this section the forward-backward and forward-only HMM simulations for comparing the BER of the Markov models with the BER of the W-CDMA simulator are discussed.

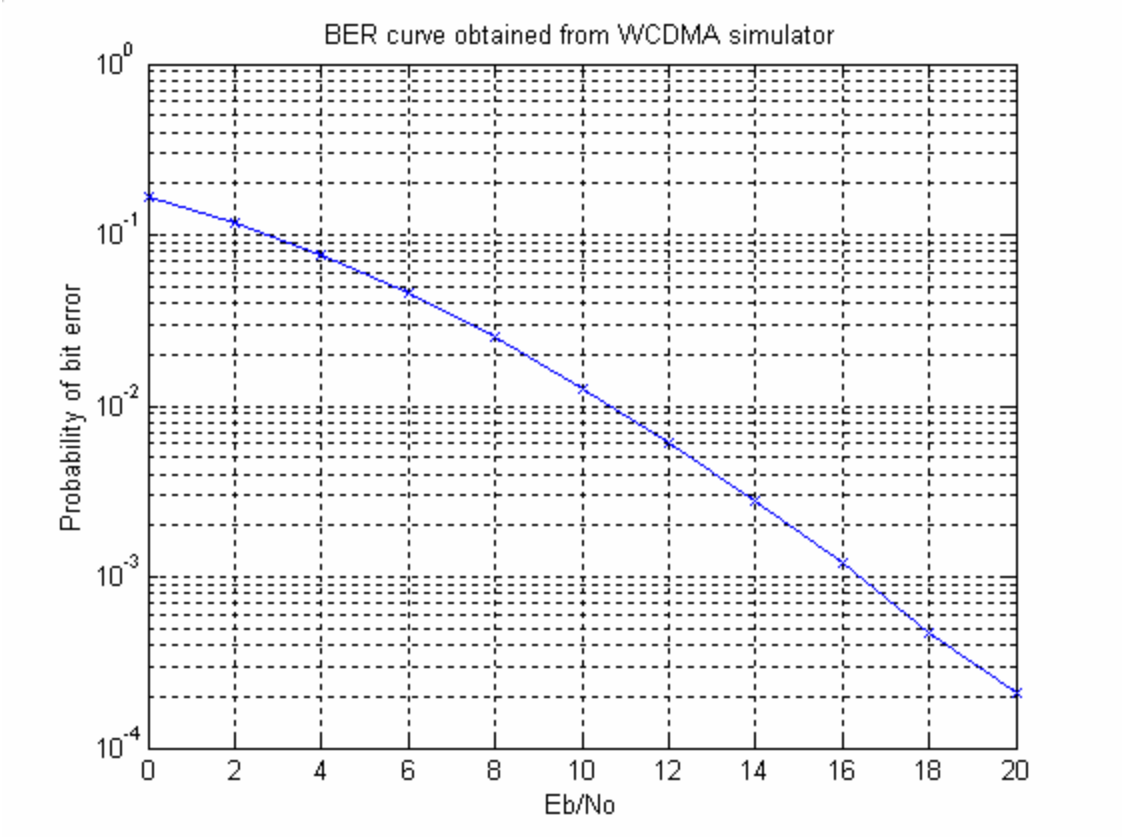

Figure 5.3.1: W-CDMA BER curve with VIPER measurement power delay profile

The power delay profiles taken from VIPER are used as input to the simulator with a noise threshold of –15 dB. Two antennas are used because our simulator is designed for two transmit antennas. The assumed velocity of the moving vehicle is set to be 50 km/hour. There are 3 interferers, 3 pilot symbols, 10 samples per chip and 2 DPDCHs. The spreading factor is set to 4. The error sequences obtained from the simulator are applied to the algorithms which are designed to estimate the parameters of Markov models.

5.3.1 Forward-Backward HMM BER Comparison

Simulations show that the runtime of the forward-backward HMM can be further reduced using vectorized MATLAB subroutines. The numerical efficiency is also increased by eliminating γ and ξ variables. We can also use thresholds for the convergence of the

model. The choice of the values of threshold is user dependent, but the threshold is normally selected having an order of 10^{-3}.

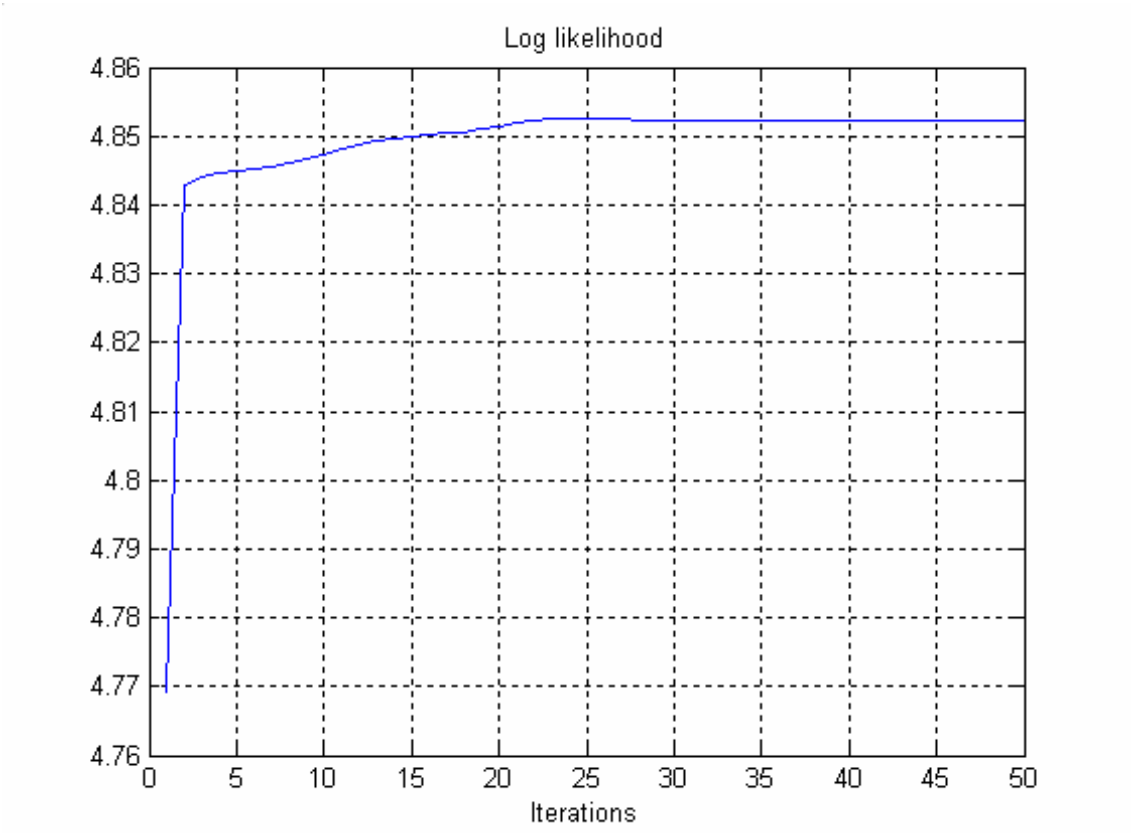

Figure 5.3.1.1: Absolute of the log-likelihood of forward-backward HMM (Eb/No 3 dB)

Eb / No (dB)	**P**	**B**	**π**
0	$\begin{bmatrix} 0.9665 & 0.0307 & 0.0028 \\ 0.0383 & 0.8616 & 0.1001 \\ 0.0370 & 0.0045 & 0.9586 \end{bmatrix}$	$\begin{bmatrix} 0.9189 & 0.7779 & 0.7266 \\ 0.0811 & 0.2221 & 0.2734 \end{bmatrix}$	$\begin{bmatrix} 0.0000 & 0.0003 & 0.9997 \end{bmatrix}$
2	$\begin{bmatrix} 0.9631 & 0.0041 & 0.0328 \\ 0.0287 & 0.9241 & 0.0472 \\ 0.0471 & 0.0220 & 0.9310 \end{bmatrix}$	$\begin{bmatrix} 0.9857 & 0.7439 & 0.8909 \\ 0.0143 & 0.2561 & 0.1091 \end{bmatrix}$	$\begin{bmatrix} 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
4	$\begin{bmatrix} 0.9631 & 0.0041 & 0.0328 \\ 0.0287 & 0.9241 & 0.0472 \\ 0.0471 & 0.0220 & 0.9310 \end{bmatrix}$	$\begin{bmatrix} 0.9857 & 0.7439 & 0.8909 \\ 0.0143 & 0.2561 & 0.1091 \end{bmatrix}$	$\begin{bmatrix} 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
6	$\begin{bmatrix} 0.9279 & 0.0632 & 0.0089 \\ 0.0380 & 0.6188 & 0.3432 \\ 0.0154 & 0.0054 & 0.9793 \end{bmatrix}$	$\begin{bmatrix} 0.8283 & 0.8747 & 0.9886 \\ 0.1717 & 0.1253 & 0.0114 \end{bmatrix}$	$\begin{bmatrix} 0.9971 & 0.0000 & 0.0029 \end{bmatrix}$
8	$\begin{bmatrix} 0.9228 & 0.0011 & 0.0761 \\ 0.0599 & 0.4021 & 0.5380 \\ 0.0117 & 0.0062 & 0.9821 \end{bmatrix}$	$\begin{bmatrix} 0.8500 & 0.9451 & 0.9948 \\ 0.1500 & 0.0549 & 0.0052 \end{bmatrix}$	$\begin{bmatrix} 1.0000 & 0.0000 & 0.0000 \end{bmatrix}$
10	$\begin{bmatrix} 0.8941 & 0.0221 & 0.0839 \\ 0.1163 & 0.6614 & 0.2223 \\ 0.0043 & 0.0077 & 0.9880 \end{bmatrix}$	$\begin{bmatrix} 0.8607 & 0.9201 & 0.9982 \\ 0.1393 & 0.0799 & 0.0018 \end{bmatrix}$	$\begin{bmatrix} 0.0000 & 0.0000 & 1.0000 \end{bmatrix}$
12	$\begin{bmatrix} 0.6328 & 0.0170 & 0.3502 \\ 0.0402 & 0.8816 & 0.0783 \\ 0.0298 & 0.0032 & 0.9670 \end{bmatrix}$	$\begin{bmatrix} 0.9812 & 0.8684 & 0.9999 \\ 0.0188 & 0.1316 & 0.0001 \end{bmatrix}$	$\begin{bmatrix} 0.0037 & 0.0000 & 0.9963 \end{bmatrix}$
14	$\begin{bmatrix} 0.9355 & 0.0269 & 0.0377 \\ 0.0575 & 0.9385 & 0.0040 \\ 0.0721 & 0.0192 & 0.9086 \end{bmatrix}$	$\begin{bmatrix} 1.0000 & 1.0000 & 0.9872 \\ 0.0000 & 0.0000 & 0.0128 \end{bmatrix}$	$\begin{bmatrix} 0.2565 & 0.7297 & 0.0138 \end{bmatrix}$
16	$\begin{bmatrix} 0.9886 & 0.0012 & 0.0102 \\ 0.0747 & 0.8212 & 0.1042 \\ 0.4124 & 0.0102 & 0.5774 \end{bmatrix}$	$\begin{bmatrix} 0.9999 & 0.8825 & 0.9914 \\ 0.0001 & 0.1175 & 0.0086 \end{bmatrix}$	$\begin{bmatrix} 0.9686 & 0.0000 & 0.0314 \end{bmatrix}$
18	$\begin{bmatrix} 0.2928 & 0.0112 & 0.6960 \\ 0.0883 & 0.8254 & 0.0863 \\ 0.0060 & 0.0002 & 0.9938 \end{bmatrix}$	$\begin{bmatrix} 0.9960 & 0.8431 & 0.9998 \\ 0.0040 & 0.1569 & 0.0002 \end{bmatrix}$	$\begin{bmatrix} 0.0259 & 0.0000 & 0.9741 \end{bmatrix}$
20	$\begin{bmatrix} 0.5802 & 0.1838 & 0.2360 \\ 0.0077 & 0.9130 & 0.0793 \\ 0.0293 & 0.1118 & 0.8589 \end{bmatrix}$	$\begin{bmatrix} 0.9944 & 1.0000 & 1.0000 \\ 0.0056 & 0.0000 & 0.0000 \end{bmatrix}$	$\begin{bmatrix} 0.0654 & 0.5552 & 0.3793 \end{bmatrix}$

Table 5.3.1.1: Forward-backward HMM simulation result

Figure 5.3.1.1 shows the absolute of the likelihood function. Note that after about 25 iterations the likelihood does not change significantly. In Figure 5.3.1.2 we show the comparison of the BER curves for the original sequence and those that were obtained from the Markov model. Different error sequences are given that were obtained from the W-CDMA simulator, for different values of E_b / N_0, as input to the forward-backward model, and thus we obtain the following plots.

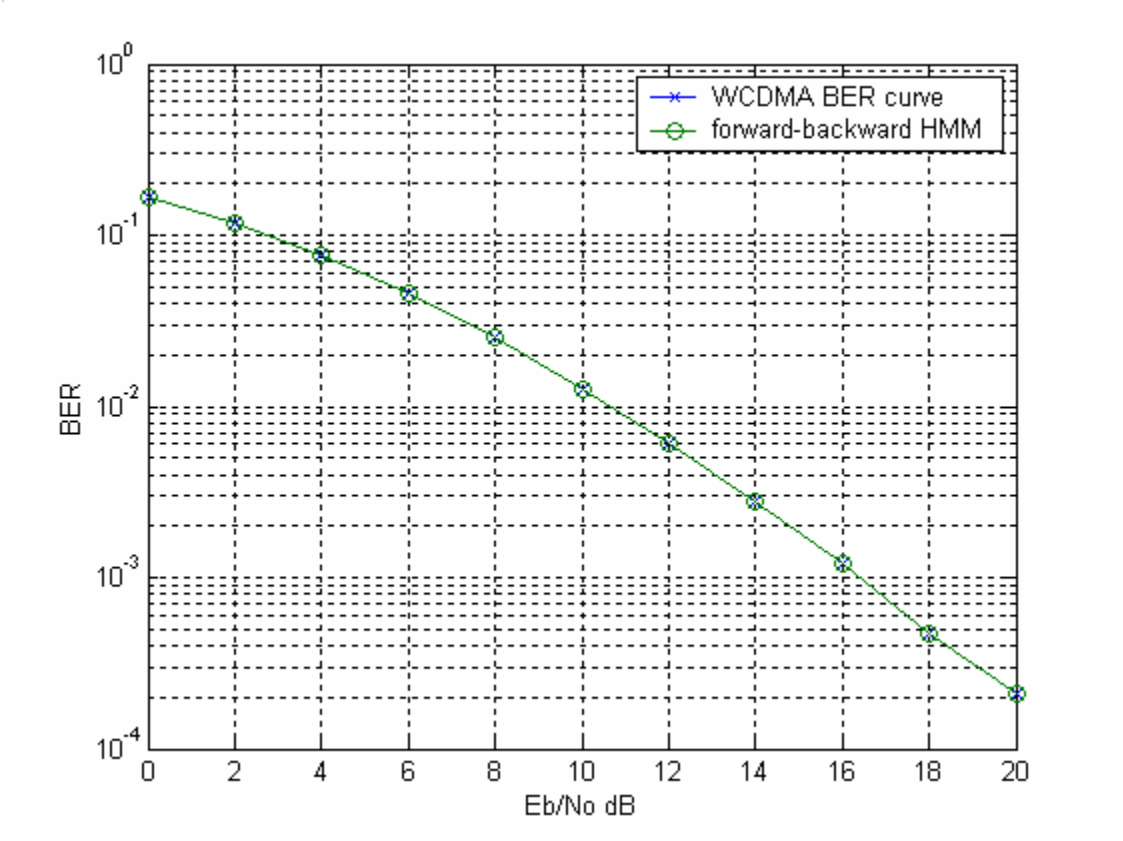

Figure 5.3.1.2: BER results for forward-backward HMM

Forward-backward HMM perfectly models the error sources obtained from the W-CDMA simulator. Table 5.3.1.1 shows the results that were obtained for different values of E_b / N_0.

5.3.2 Forward-Only HMM Simulations

In this section we analyze the fitting of the W-CDMA error vector to the forward-only HMM and obtain the results as shown in Figure 5.3.2.1

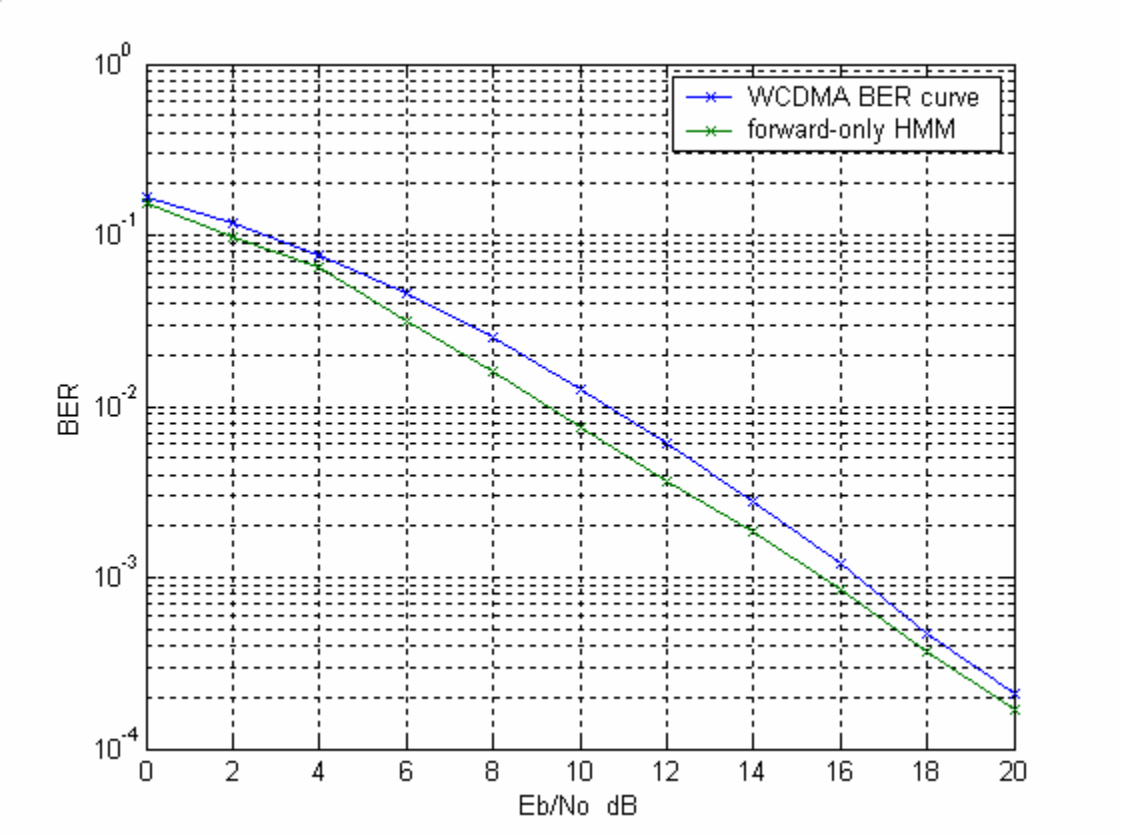

Figure 5.3.2.1: BER results for forward-only HMM

Comparison between BER results using forward-only HMM indicates that this algorithm is not as accurate as its basic version, i.e. forward-backward HMM. Table 5.3.2.1 shows the results that were found for different values of E_b / N_0.

Eb / *No* (dB)	P	B
0	$\begin{bmatrix} 0.8763 & 0.1156 & 0.0081 \\ 0.0313 & 0.8721 & 0.0966 \\ 0.1364 & 0.0035 & 0.8601 \end{bmatrix}$	$\begin{bmatrix} 0.8471 & 0.8469 & 0.8458 \\ 0.1529 & 0.1531 & 0.1542 \end{bmatrix}$
2	$\begin{bmatrix} 0.8737 & 0.1180 & 0.0083 \\ 0.0155 & 0.9784 & 0.0061 \\ 0.0107 & 0.0687 & 0.9206 \end{bmatrix}$	$\begin{bmatrix} 0.8930 & 0.9049 & 0.8930 \\ 0.1070 & 0.0951 & 0.1070 \end{bmatrix}$
4	$\begin{bmatrix} 0.8719 & 0.1193 & 0.0087 \\ 0.0299 & 0.8692 & 0.1009 \\ 0.1256 & 0.0034 & 0.8710 \end{bmatrix}$	$\begin{bmatrix} 0.9395 & 0.9393 & 0.9382 \\ 0.0605 & 0.0607 & 0.0618 \end{bmatrix}$
6	$\begin{bmatrix} 0.9052 & 0.0806 & 0.0142 \\ 0.0254 & 0.9616 & 0.0129 \\ 0.0093 & 0.0299 & 0.9608 \end{bmatrix}$	$\begin{bmatrix} 0.9645 & 0.9702 & 0.9676 \\ 0.0355 & 0.0298 & 0.0324 \end{bmatrix}$
8	$\begin{bmatrix} 0.8655 & 0.0430 & 0.0915 \\ 0.2031 & 0.7356 & 0.0613 \\ 0.0433 & 0.0432 & 0.9134 \end{bmatrix}$	$\begin{bmatrix} 0.9839 & 0.9820 & 0.9846 \\ 0.0161 & 0.0180 & 0.0154 \end{bmatrix}$
10	$\begin{bmatrix} 0.9426 & 0.0532 & 0.0042 \\ 0.0702 & 0.8270 & 0.1028 \\ 0.2510 & 0.0027 & 0.7463 \end{bmatrix}$	$\begin{bmatrix} 0.9927 & 0.9921 & 0.9912 \\ 0.0073 & 0.0079 & 0.0088 \end{bmatrix}$
12	$\begin{bmatrix} 0.9321 & 0.0666 & 0.0013 \\ 0.0336 & 0.9646 & 0.0017 \\ 0.0701 & 0.1995 & 0.7304 \end{bmatrix}$	$\begin{bmatrix} 0.9963 & 0.9964 & 0.9944 \\ 0.0037 & 0.0036 & 0.0056 \end{bmatrix}$
14	$\begin{bmatrix} 0.8753 & 0.0309 & 0.0939 \\ 0.2577 & 0.6634 & 0.0789 \\ 0.0434 & 0.0307 & 0.9259 \end{bmatrix}$	$\begin{bmatrix} 0.9981 & 0.9981 & 0.9981 \\ 0.0019 & 0.0019 & 0.0019 \end{bmatrix}$
16	$\begin{bmatrix} 0.9301 & 0.0480 & 0.0219 \\ 0.0209 & 0.5712 & 0.4079 \\ 0.0088 & 0.0050 & 0.9862 \end{bmatrix}$	$\begin{bmatrix} 0.9991 & 0.9991 & 0.9992 \\ 0.0009 & 0.0009 & 0.0008 \end{bmatrix}$
18	$\begin{bmatrix} 0.8468 & 0.0026 & 0.1506 \\ 0.0697 & 0.5927 & 0.3376 \\ 0.0320 & 0.0182 & 0.9499 \end{bmatrix}$	$\begin{bmatrix} 0.9996 & 0.9996 & 0.9996 \\ 0.0004 & 0.0004 & 0.0004 \end{bmatrix}$
20	$\begin{bmatrix} 0.1527 & 0.0998 & 0.7475 \\ 0.0076 & 0.8306 & 0.1619 \\ 0.0016 & 0.0258 & 0.9726 \end{bmatrix}$	$\begin{bmatrix} 0.9998 & 0.9998 & 0.9998 \\ 0.0002 & 0.0002 & 0.0002 \end{bmatrix}$

Table 5.3.2.1: Forward-only HMM simulation result

5.4 Semi-Hidden Markov Model Simulations

This section discusses the semi-Markov model simulations for the W-CDMA environment. Both forward-backward and forward-only semi-Markov models are tested and their results are shown.

5.4.1 Forward-Backward Semi-Markov Model Simulations

The results of a forward-backward semi-hidden Markov model having one good state and two bad states are shown. The **B** matrix, in this particular case, is assumed to take the form as shown below

$$\mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \end{bmatrix}$$

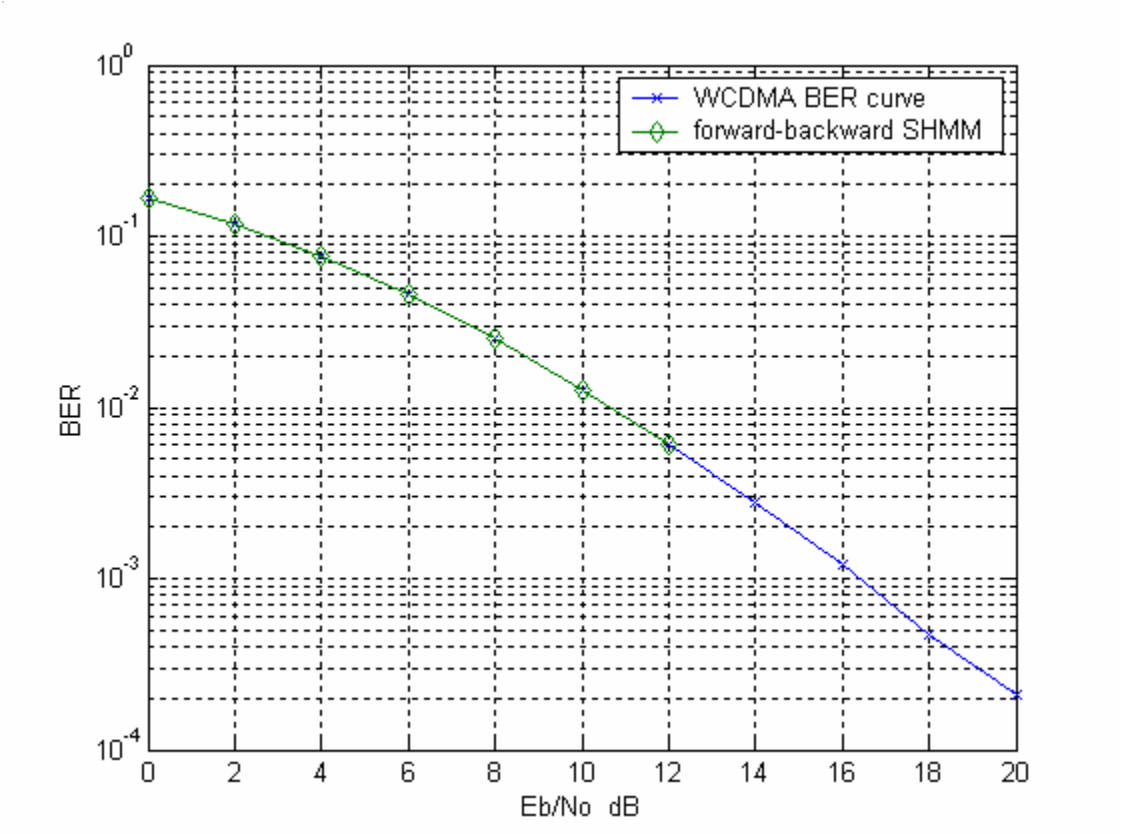

Figure 5.4.1.1: BER results for forward-backward semi-HMM

From the BER comparison of forward-backward HMM we find that forward-backward HMM perfectly models the error sequences obtained from the W-CDMA simulator, with much faster estimation runtime. Table 5.4.1.1 shows the forward-backward SHMM results for different values of the E_b / N_0.

Eb / *No* (dB)	**P**
0	$\begin{bmatrix} 0.8763 & 0.1156 & 0.0081 \\ 0.0313 & 0.8721 & 0.0966 \\ 0.1364 & 0.0035 & 0.8601 \end{bmatrix}$
2	$\begin{bmatrix} 0.8906 & 0.0744 & 0.0350 \\ 0.8839 & 0.1161 & 0 \\ 0.7166 & 0 & 0.2834 \end{bmatrix}$
4	$\begin{bmatrix} 0.9288 & 0.0584 & 0.0128 \\ 0.8928 & 0.1072 & 0 \\ 0.6967 & 0 & 0.3033 \end{bmatrix}$
6	$\begin{bmatrix} 0.9578 & 0.0195 & 0.0227 \\ 0.7922 & 0.2078 & 0 \\ 0.9576 & 0 & 0.0424 \end{bmatrix}$
8	$\begin{bmatrix} 0.9770 & 0.0089 & 0.0142 \\ 0.7933 & 0.2067 & 0 \\ 0.9547 & 0 & 0.0453 \end{bmatrix}$
10	$\begin{bmatrix} 0.9885 & 0.0082 & 0.0032 \\ 0.9488 & 0.0512 & 0 \\ 0.7857 & 0 & 0.2143 \end{bmatrix}$
12	$\begin{bmatrix} 0.9944 & 0.0053 & 0.0003 \\ 0.9364 & 0.0636 & 0 \\ 0.6146 & 0 & 0.3854 \end{bmatrix}$

Table 5.4.1.1: Forward-backward SHMM simulation result

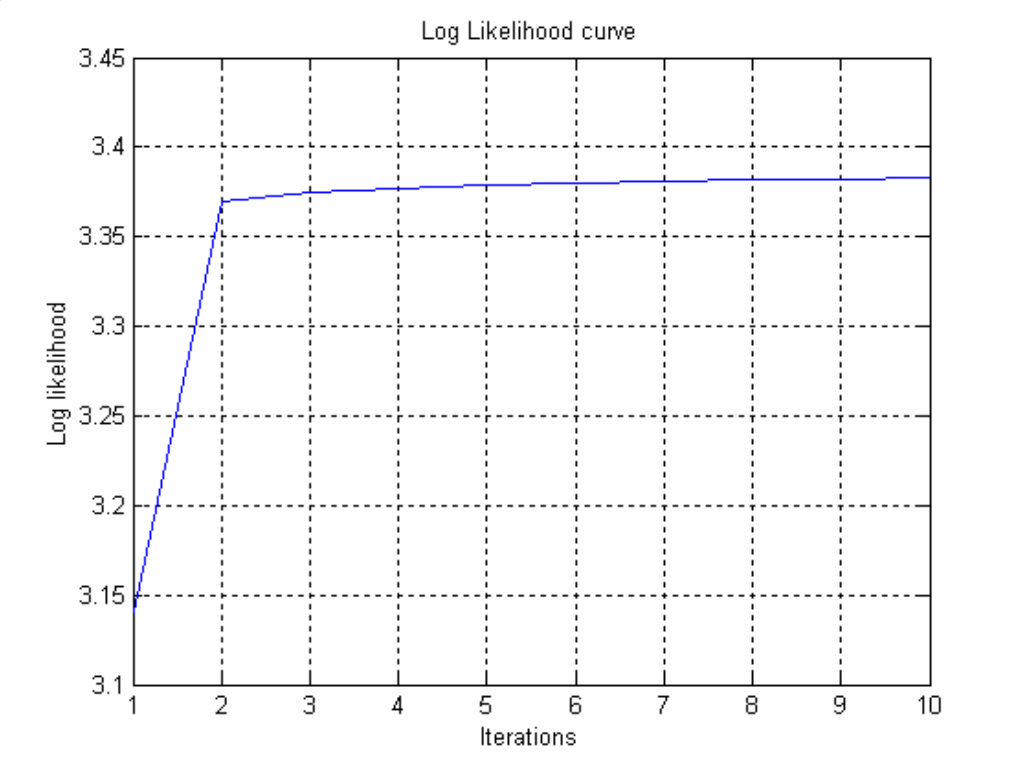

Figure 5.4.1.2: Absolute of log likelihood for forward-backward SHMM (Eb/No 3 dB)

Figure 5.4.1.2 shows the convergence of the absolute of the log likelihood function for different iterations with E_b / N_0 of 3 dB. The algorithm finds the local maxima and does not change considerably after some initial iterations.

5.4.2 Forward-Only Semi-Markov Model Simulations

In this section the simulation results of the forward-only semi-HMM with one good state and two bad states are shown. Results obtained from this model are shown in Figure 5.4.2.1

Eb / *No* **(**dB**)**	**P**
0	$\begin{bmatrix} 0.6326 & 0.1837 & 0.1837 \\ 0.9627 & 0.0373 & 0 \\ 0.9627 & 0 & 0.0373 \end{bmatrix}$
2	$\begin{bmatrix} 0.7382 & 0.1309 & 0.1309 \\ 0.9796 & 0.0204 & 0 \\ 0.9796 & 0 & 0.0204 \end{bmatrix}$
4	$\begin{bmatrix} 0.8305 & 0.0847 & 0.0847 \\ 0.9898 & 0.0102 & 0 \\ 0.9898 & 0 & 0.0102 \end{bmatrix}$
6	$\begin{bmatrix} 0.9016 & 0.0492 & 0.0492 \\ 0.9953 & 0.0047 & 0 \\ 0.9953 & 0 & 0.0047 \end{bmatrix}$
8	$\begin{bmatrix} 0.9473 & 0.0264 & 0.0264 \\ 0.9978 & 0.0022 & 0 \\ 0.9978 & 0 & 0.0022 \end{bmatrix}$
10	$\begin{bmatrix} 0.9742 & 0.0129 & 0.0129 \\ 0.9991 & 0.0009 & 0 \\ 0.9991 & 0 & 0.0009 \end{bmatrix}$

Table 5.4.2.1: Forward-only SHMM simulation result

Table 5.4.2.1 shows the estimated values of the models that were obtained for different values of E_b / N_0.

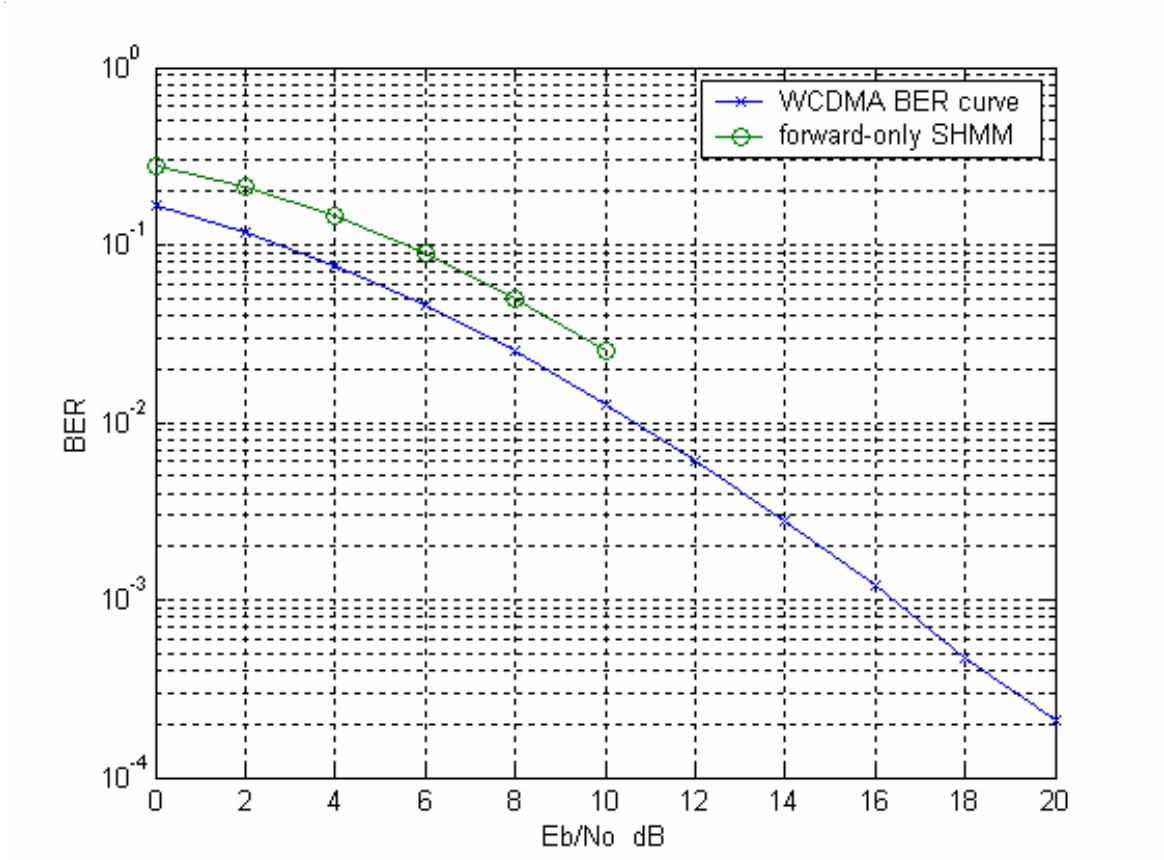

Figure 5.4.2.1: BER results for forward-only semi-HMM

Figure 5.4.2.1 shows the BER comparison for forward-only semi-HMMs. The curves do not match closely in this case.

5.5 Conclusion

As a conclusion, it can be stated that we can use forward-backward HMMs and semi-HMMs to efficiently model error sources obtained from the W-CDMA simulator. Forward-only HMMs are very useful for real-time applications because they only use the forward-path. These forward-only algorithms are designed for both hidden Markov models and semi-hidden Markov models. Forward-only semi-HMMs however sometime do not model these error sources accurately. However, forward-backward Markov models (hidden Markov models and semi-hidden Markov models) can model the error statistics of the W-CDMA channel very accurately. Accuracy of forward-backward HMMs and

forward-backward SHMMs are almost the same and both models can be used confidently in third generation wireless channels simulations. These models however are computationally inefficient and often take longer simulation runtime as compared to their forward-only counterparts.

We also conclude from this thesis that state duration distributions for both flat fading and frequency selective channels can be modeled using discrete channel models.

We have thus implemented different Markov models that can replace the waveform level simulations of the W-CDMA simulator.

Bibliography

[1] R. A. Howard, *Dynamic Probabilistic Systems, Volume I: Markov Models*, John Wiley and Sons Inc., 1971.

[2] R. A. Howard, *Dynamic Probabilistic Systems, Volume. II: SemiMarkov and Decision Processes*, John Wiley and Sons Inc., 1971.

[3] C. E. Shannon, "A Mathematical Theory of Communications," *Bell Syst. Tech. J.* vol. 27, July, October, 1948, pp. 379-423, 623-656. Also in *Claude Elwood Shannon Collected Papers*, N. J. A. Sloan and A. D. Wyner Eds, IEEE Press, Piscataway, New Jersey, 1993.

[4] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Syst. Tech. J.*, vol. 39, Sept. 1960, pp. 1253-1266.

[5] B. D. Fritchman, "A binary characterization using partitioned Markov chains," *IEEE Trans. Inform. Theory*, vol. IT-13, Apr. 1967, pp. 221-227.

[6] W. Turin, *Digital Transmission Systems: Performance Analysis and Modeling*, McGraw-Hill, 1998.

[7] E. Cinlar, *Introduction to Stochastic Processes*, Prentice Hall, Englewood Cliffs, New Jersey, 1975.

[8] W. Feller, *An introduction to Probability Theory and Its Applications,* vol. 1, John Wiley & Sons, New York, 1962.

[9] W. Jakes, *Microwave Mobile Communications,* John Wiley & Sons, New York, 1974.

[10] J. G. Proakis, *Digital Communications*, McGraw-Hill, New York, 1989.

[11] L. Rabiner and B.-H. Jaung, *Fundamentals of Speech Recognition,* Prentice Hall, Englewood Cliffs, New Jersey, 1993.

[12] S. Sivaprakasam and K. S. Shanmugan, "An Equivalent model for burst errors in digital channels," *IEEE Trans. Comm.*, vol. 43, April 1995, pp. 1345-1355.

[13] W. Turin, and M. M. Sondhi, "Modeling error sources in digital channels," *IEEE Journ. Sel. Areas in Communications*, vol. 11, No. 3. 1993, pp. 340-347.

[14] W. Turin and R. Van Nobelen, "Hidden Markov modeling of fading channels," *48th Veh., Tech. Conf.*, May, 1998, pp. 1234-1238.

[15] H. S. Wang and N. Moayeri, "Finite-state Markov channels – A useful model for radio communications channels," *IEEE Trans. Veh. Tech.,*vol. 44. Feb. 1995, pp. 163-171.

[16] H. S. Wang and P.-C. Chang, "On verifying the first-order Markovian assumption for a Rayleigh fading channel model," *IEEE Trans. Veh. Tech.*, vol. 45, May 1996, pp. 353-357.

[17] M. Zorzi and R. R. Rao, "On the statistics of block errors in bursty channels," *IEEE Trans. Comm.,* vol. 45, Jun. 1997, pp. 660-667.

[18] F. R. Gantmacher, *The Theory of Matrices,* Chelsea Publishing Co., New York, 1959.

[19] M. Sajadieh and F. R. Kschischang, and A. Leon-Garcia, "A block memory model for correlated Rayleigh fading channels." *Roc. IEEE Int. Conf. Comm.,* June 1996, pp. 282-286.

[20] S. Sivaprakasam and K. S. Shanmugan, "An equivalent model for burst errors in digital channels," GLOBECOM, 1995.

[21] Cecilo Pimentel and Ian F. Blake, "Modeling Burst Channels Using Partitioned Fritchman's Markov Models," *IEEE Trans. Veh. Tech.,* vol. 47, no. 3, August 1998.

[22] L. R. Rabiner and B. H. Juang, "An introduction to Hidden Markov Models," *IEEE ASSP Magazine,* January 1986.

[23] Leonard E. Baum, Ted Petrie, George Soules and Norman Weiss, "A Maximization Technique Occurring in the Statistical Analysis of Probabilistic Functions of Markov Chains," *Annals of Mathematical Statistics,* vol. 41, Issue 1 (Feb., 1970), pp. 164-171.

[24] Mordechai Mushkin and Israel Bar-David, "Capacity and Coding for the Gilbert-Elliott Channels," *IEEE Trans. Inform. Theory,* vol. 35, no. 6, November 1989.

[25] Javier Garcia-Frias and John D. Villasenor, "Exploiting Binary Markov Channels With Unknown Parameters in Turbo Codes," *Proc. Globecom,* pp. 3244-3249, Sydney, Australia, November 1998.

[26] Andrea J. Goldsmith and Pravin P. Varaiya, "Capacity, Mutual Information, and Coding for Finite-State Markov Channels."

[27] Paul M. Baggenstoss, "A Modified Baum-Welch Algorithm for Hidden Markov Models with Multiple Observation Spaces", *IEEE Trans. Speech and Audio Processing,* vol. 9, May 2001.

[28] T. Hasegawa, H. Yasuda and T. Matsumoto, "Fast Discrete HMM Algorithm for On-line Handwriting Recognition."

[29] Jamie S. Evans and Vikram Krishnamurthy, "Hidden Markov Model State Estimation with Randomly Delayed Observations," *IEEE Trans. Signal Processing,* vol. 47, no. 8, August 1998.

[30] Javier Garcia-Frias and John D. Villasenor, "Combining Hidden Markov Source Models and Parallel Concatenated Codes," *IEEE Comm. Letters,* pp. 111-113, July 1997.

[31] Javier Garcia-Frias and John D. Villasenor, "Turbo Codes for Binary Markov Channels," *Proc ICC,* pp. 110-115, Atlanta, GA, June 1998.

[32] William Turin and Robert van Nobelen, "Hidden Markov Modeling of Flat Fading Channels," *IEEE Journ. Selected Areas Comm.,* vol. 16, no. 9, December 1998.

[33] Michele Zorzi and Ramesh R. Rao, "On the Statistics of Block Errors in Bursty Channels," *accepted for publication in the IEEE Trans. Comm.*

[34] Michele Zorzi, Ramesh R. Rao and Laurence B. Milstein, "A Markov Model for Block Errors on Fading Channels," *PIMRC'96,* Taiwan, Oct. 1996.

[35] Fulvio Babich, Owen E. Kelly and Giancarlo Lombardi, "Generalized Markov Modeling for Flat Fading," *IEEE Trans. Comm.,* vol. 48, no. 4, April 2000.

[36] Tapas Knungo, "Hidden Markov Models," *www.cfar.umd.edu/~kanungo.*

[37] S. Sivprakasam and K. S. Shanmugan, "A Forward-Only Recursion based HMM for Modeling Burst Errors in Digital Channels," *1995 IEEE.*

[38] E. O. Elliott, "Estimates of error rates for codes on burst-noise channels," *Bell Syst. Tech. Journ.,* vol. 42, Sept. 1963, pp. 1977-1997.

[39] W. C. Y. Lee, *Mobile Communications Engineering,* 2nd ed., McGraw-Hill, New York, 1997.

[40] S. O. Rice, "Distribution of duration of fades in radio transmission: Gaussian noise model," *Bell Syst. Tech. Journ.,* vol. 37, May 1958, pp. 581-635.

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

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

[43] J. Garcia-Frias and P.M. Crespo, "Hidden Markov models for burst error characterization in indoor radio channels," *IEEE Trans. Veh. Tech*, vol. 46, no. 4, pp. 1006-1019, November 1997.

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

[45] R.H. Clarke, "A statistical theory of mobile radio reception," *Bell Syst. Tech Journ.*, vol. 47, pp. 957-1000, 1968.

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

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

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

[49] H. Kong, and E. Shwedyk, "Sequence detection and channel state estimation over finite state Markov channels," *IEEE Transac. Veh. Tech*, vol. 48, no. 3, pp. 833-839, May 1999.

[50] Ford, and Moore, "Adaptive Estimation of HMM Transition Probabilities," *IEEE Transac. Signal Processing*, vol. 46, No. 5, May 1998.

[51] William Turin, "Probability Distribution Laws for the Numbers of Errors in Several Combinations," *Telecomm. and Radio Engg.*, vol. 28, no. 12 Dec. 1973.

[52] Kemeney, Snell, *Finite Markov Chains*, D. Van Nostrand Co., Princeton, NJ, 1960.

[52] Billingsley, Patrick, *Statistical Inference for Markov Processes*, University of Chicago Press, Chicago, IL, 1961.

[53] Lee, Judge, Zellner, *Estimating the Parameters of the Markov Probability Model from Aggregate Time Series Data*, North-Holland Publishing Company, London, England, 1970.

[54] Roger A. Horn and Charles R. Johnson, *Matrix Analysis*, Cambridge University Press, 1985.

[55] Marius Iosifescu, *Finite Markov Processes and Their Applications*, John Wiley & Sons, 1980.

[56] Dean L. Issacson and Richard W. Madsen, *Markov Chains Theory and Applications,* John Wiley & Sons, 1976.

[57] Masaaki Kijima, *Markov Processes for Stochastic Modeling*, Chapman & Hall, 1997.

[58] Vladmir Kalashnikov, *Topics on Regenerative Processes,* Library of Congress Cataloging-in-Publication Data, 1994.

[59] Korolyuk and A. Swishchuk, *Semi-Markov Random Evaluations,* Kluwer Academic Publishers, 1995.

[60] J. R. Norris, *Markov Chains,* Cambridge University, 1997.

[61] Papoulis, A., *Probability, Random Variables, and Stochastic Processes*, McGraw-Hill, 3rd. Edition, 1991.

[62] Bremaud, P., *An Introduction to Probabilistic Modeling*, NY: Springer Verlag, 1988.

[63] Grimmett, G.R. and Stirzaker, D.R., *Probability and Random Processes*, Oxford Science Publications, 1998.

[64] Bremaud, P., *An Introduction to Probabilistic Modeling*, NY: Springer Verlag, 1988.

[65] Ross, S. M., *Introduction to Probability Models*, Academic Press, Inc., 1993.

[66] Wong, E. and B. Hajek, *Stochastic Processes in Engineering Systems*, Springer Verlag, 1985.

[67] Ross, S. , *A First Course in Probability*, Prentice Hall, 5th. Edition,1998

[68] M. C. Jeruchim, Philip Balaban and K. S. Shanmugan, *Simulation of Communication Systems,* Plenum Press, New York, 1992.

[69] William Turin, *Performance of Digital Transmission Systems,* Computer Science Press, 1990.

[70] Vijay K. Garg and Joseph E. Wikes, *Wireless Personal Communications,* Prentice Hall, 1996.

[71] Rodger E. Ziemer and Roger L. Peterson, *Introduction to Digital Communications,* 2nd edition, Prentice Hall 2001.

[72] Raqibul Mostafa, Ran Gozali, William G. Newhall, Ihsan Akbar, Dr. Jeffrey H. Reed, Dr. Brian D. Woerner and Dr. William H. Tranter, "Navy Collaborative Integrated Information Technology Initiative, Report #15," *MPRG-TR-01-11, Technical Report*, July 31, 2001.

[73] Newhall W.G., *Radio Channel Modeling and Measurements for Wideband, Multi-Element Antenna Communication Systems*, Ph.D. Dissertation, Bradley Department of Electrical and Computer Engineering, Virginia Tech, Blacksburg, Virginia.

[74] A. P. Dumper, N. M. Laird and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *J. R. Statist.,* vol. 76, 1977, pp 341-353.

[75] S. Kullback, *Information Theory and Statistics,* John Wiley & Sons, New York 1959.

[76] Ross F. Barrett and David A. Holdsworth, "Frequency Tracking Using Hidden Markov Models with Amplitude and Phase Information," *IEEE Transactions on Signal Processing,* vol. 41, no. 10, October 1993.

[77] Vikram Krishnamurthy and John B. Moore, "On-Line Estimation of Hidden Markov Model Parameters Based on the Kullback-Leibler Information Measure," *IEEE Transactions on Signal Processing,* vol. 41, no. 8, August 1993.

[78] Helmut Lucke, "Which Stochastic Models Allow Baum-Welch Training?," *IEEE Transactions on Signal Processing,* vol. 44, no. 11, November 1996.

[79] J. Connan and C. W. Omlin, "Bibliography Extraction with Hidden Markov Models," Department of Computer Science, University of Stellenbosch, February 2000.

[80] Volker Breuer and Gunter Rados, "Baum-Welch Reestimation Formula for Coupled Sequential Machines," *IEEE Trans. Signal Proc.,* vol. 43, no. 5, May 1995.

[81] Carles Antón-Haro, José A. R. Fonollosa and Javier R. Fonollosa, "Blind Channel Estimation and Data Detection Using Hidden Markov Models," *IEEE Trans. Signal Proc.,* vol. 45, no. 1, January 1997.

[82] Iain B. Collings, Vikran Krishnamurthy and John B. Moore, "On-Line Identification of Hidden Markov Models via Recursive Prediction Error Techniques," *IEEE Trans. Signal Proc.,* vol. 42, no. 12, December 1994.

[83] Micheal J. Chu, Dennis L. Goeckel and Wayne E. Stark, "On the Design of Markov Models for Fading Channels."

[84] P. M. Soni and A. Chokalingam, "Analysis of Link Layer Backoff Schemes on Point-to-Point Markov Fading Links," *Tech. Report WRL-IISc-TR-101.*

[85] P. M. Soni and A. Chokalingam, "Performance Analysis of UDP with Energy Efficient Link Layer on Markov Fading Channels," *http://wrl.ece.iisc.emet.in*

[86] Iain B. Collings and John B. Moore, "Adaptive HMM filters for Signals in Noisy Fading Channels," *Dept. of Systems Engineering, Research School of Information Sciences and Engineering,* Australian National University, Canberra 0200, Australia.

[87] J. K. Romberg, H. Choi and R. G. Baraniuk, "Bayesian Tree Structured Image Modeling using Wavelet-domain Hidden Markov Models," *IEEE Transactions on Image Processing*, March 2000.

[88] Matthew S. Crouse, Robert D. Nowark and Irchard G. Baranuik, "Wavelet-Based Statistical Signal Processing Using Hidden Markov Models," *IEEE Trans. Signal Proc., Special issue on wavelets and filter banks,* April 1998.

[89] M. Jaber and Robert D. Nowak, "Wavelet-Based Denoising Using Hidden Markov Models," *ECE Dept., Rice University.*

Appendix

Here, we drive the forward backward BWA for estimating the parameters of HMMs. We introduce two new variables γ and ξ during this derivation, but they are not required in the actual implementation. Optimized MATLAB code is used for implementing this algorithm that takes relatively less computer memory and runs very fast as compared to the traditional BWA [11], [22]. Let's continue from (3.5.13) and define

$$y_t(i,j) = \begin{Bmatrix} 1, \text{ at time } t, \text{ the system transitions from } s_i \text{ to } s_j \\ 0, \text{ otherwise} \end{Bmatrix} \tag{A.1}$$

and

$$y(i,j) = \text{number of transitions from state } s_i \text{ to } s_j \tag{A.2}$$

so that

$$y(i,j) = \sum_{t=1}^{T\text{-}1} y_t(i,j) \tag{A.3}$$

Let us also define

$$y_t(i) = \begin{Bmatrix} 1, \text{ at time } t, \text{ the system transitions from state } s_j \\ 0, \text{ otherwise} \end{Bmatrix} \tag{A.4}$$

and

$$y(i) = \text{number of transitions from state } s_i \tag{A.5}$$

so that

$$y(i) = \sum_{t=1}^{T-1} y_t(i) \tag{A.6}$$

We also define the variable

$$z_t(j,k) = \left\{ \begin{array}{l} 1, \text{ at time } t, \text{ the system is in state } s_j \text{ with output observable } v_k \\ 0, \text{ otherwise} \end{array} \right\} \tag{A.7}$$

and

$$z(j,k) = \text{number of times in state } s_j \text{ and observing symbol } v_k \tag{A.8}$$

so that

$$z(j,k) = \sum_{t=1}^{T} z_t(j,k) \tag{A.9}$$

and also

$$z_t(j) = \left\{ \begin{array}{l} 1, \text{ at time } t, \text{ the system is in state } s_j \\ 0, \text{ otherwise} \end{array} \right\} \tag{A.10}$$

$$z(j) = \text{number of times in state } s_j \tag{A.11}$$

so that

$$z(j) = \sum_{t=1}^{T} z_t(j) \tag{A.12}$$

The estimated values of the new model τ_{p+1} given the previous model τ_p are

$$p_{ij} = \frac{\text{expected number of transitions from state } s_i \text{ to state } s_j}{\text{expected number of transitions from state } s_i} \tag{A.13}$$

$$b_j(k) = \frac{\text{expected number of times in state } s_j \text{ and observing symbol } v_k}{\text{expected number of times in state } s_j} \tag{A.14}$$

and

$$\pi_i = \text{expected number of times in state } s_i \text{ at times } t = 1 \tag{A.15}$$

If $E[\ .\ ]$ denotes the expectation operator, then we can write

$$p_{ij} = \frac{E[y(i,j)]}{E[y(i)]} \tag{A.16}$$

$$b_j(k) = \frac{E[z(j,k)]}{E[z(i)]} \tag{A.17}$$

$$\pi_i = E[y_1(i)] \tag{A.18}$$

We know that

$$E\left[y(i,j)\right] = E\left[\sum_{t=1}^{T\text{-}1} y_t(i,j)\right] = \sum_{t=1}^{T\text{-}1} E\left[y_t(i,j)\right] \tag{A.19}$$

$$E\left[y(i,j)\right] = \sum_{t=1}^{T\text{-}1}\left[0.\Pr(y_t(i,j)=0) + 1.\Pr(y_t(i,j)=1)\right] \tag{A.20}$$

$$E\left[y(i,j)\right] = \sum_{t=1}^{T\text{-}1}\left[1.\Pr(y_t(i,j)=1)\right] \tag{A.21}$$

Let us also define

$$\xi_t(i,j) \equiv E\left[y_t(i,j)\right] \tag{A.22}$$

$$\xi_t(i,j) = \Pr\left(y_t(i,j)=1\right) = \Pr\left(q_t = s_i, q_{t+1} = \boldsymbol{s}_j \mid \boldsymbol{x}, \tau\right) \tag{A.23}$$

$$\xi_t(i,j) = \frac{\Pr\left(q_t = s_i, q_{t+1} = \boldsymbol{s}_j, \boldsymbol{x} \mid \tau\right)}{P(\boldsymbol{x} \mid \tau)} \tag{A.24}$$

This variable can be computed in terms of forward and backward variable

$$\alpha_t(i) = \Pr(x_1, x_2, ..., x_t, q_t = S_i \mid \tau) \tag{A.25}$$

$$\beta_{t+1}(j) = \Pr(x_{t+2}, x_{t+3}, ..., x_T \mid q_{t+1} = s_j, \tau) \tag{A.26}$$

This results in

$$\xi_t(i,j) = \frac{\alpha_t(i)\, p_{ij}\, b_j(x_{t+1})\beta_{t+1}(j)}{P(\boldsymbol{x} \mid \tau)} \tag{A.27}$$

The forward variable $\alpha_t(i)$ and the backward variable $\beta_{t+1}(j)$ account for the entire observation sequence $\boldsymbol{x}$ except for the observation x_{t+1} at time $t+1$, and the transition from the state s_i to s_j from t to $t+1$. These are taken care of by multiplication of $b_j(x_{t+1})$ and p_{ij}, respectively. Note that in the implementation of this algorithm, we do not have to keep the whole $\xi_t(i, j)$ to estimate the parameter of the HMM.

From the definition of

$$\sum_{i=1}^{N}\sum_{j=1}^{N}\xi_t(i,j)=1 \tag{A.28}$$

So it must be true that

$$\Pr(\boldsymbol{x}\mid\tau)=\sum_{i=1}^{N}\sum_{j=1}^{N}\alpha_t(i)\,p_{ij}b_j(x_{t+1})\beta_{t+1}(j) \tag{A.29}$$

We know that

$$\Pr(\boldsymbol{x}\mid\tau)=\sum_{i=1}^{N}\alpha_t(i)\beta_t(i) \qquad t\in\{1,2,...,T\} \tag{A.30}$$

and also

$$\beta_t(i)=\sum_{j=1}^{N}a_{ij}\mathrm{b}_j(x_{t+1})\beta_{t+1}(j), \qquad t=\{T-1,T-2,...,1\},\ i=\{1,2,...,N\} \tag{A.31}$$

and this is consistent with the equation

$$\Pr(\boldsymbol{x}\mid\tau)=\sum_{i=1}^{N}\alpha_t(i)\beta_t(i) \qquad t\in\{1,2,...,T\} \tag{A.32}$$

Now consider

$$E[y(i)]=\sum_{t=1}^{T}E\left[y_i(i)\right]=\sum_{t=1}^{T\text{-}1}\Pr\left(y_t(i)=1\right)=\sum_{t=1}^{T\text{-}1}\Pr\left(q_t=s_t\mid\tau\right) \tag{A.33}$$

Let us define $\gamma_t(i)$ as

$$\gamma_t(i) \equiv E\left[y_t(i)\right] \tag{A.34}$$

so that

$$\gamma_t(i) = \Pr\left(q_t = s_i \mid \boldsymbol{x}, \tau\right) = \sum_{j=1}^{N} \Pr\left(q_t = s_i, q_{t+1} = s_j \mid \boldsymbol{x}, \tau\right) \tag{A.35}$$

$$\gamma_{t_p}(i) = \sum_{j=1}^{N} \xi_{t_p}(i, j) \tag{A.36}$$

where the suffix p denotes the value at the pth iteration.

Hence we can write

$$p_{ij_{p+1}} = \frac{\sum_{t=1}^{T-1} \xi_{t_{p+1}}(i, j)}{\sum_{t=1}^{T-1} \gamma_{t_{p+1}}(i)} \tag{A.37}$$

$$E[z(i, j)] = \sum_{i=1}^{T} \Pr\left(z_t(j,k) = 1\right) = \sum_{t=1}^{T} \Pr(q_t = s_j, x_t = v_k \mid \boldsymbol{x}, \tau) \tag{A.38}$$

$$E\left[z(j,k)\right] = \sum_{\substack{t=1 \\ \forall x_t = v_k}}^{T} \gamma_t(j) \tag{A.39}$$

Similarly,

$$E\left[z(j)\right] = \sum_{t=1}^{T} \Pr\left(z_t(j) = 1\right) \tag{A.40}$$

$$E\left[z(j)\right] = \sum_{t=1}^{T} \gamma_t(j) \tag{A.41}$$

so that

$$b_{j_{p+1}}(k) = \frac{\sum_{\substack{t \\ \forall x_t = v_k}}^{T} \gamma_{t_p}(j)}{\sum_{t=1}^{T} \gamma_{t_p}(j)} \tag{A.42}$$

Finally, we get

$$\pi_{i_{p+1}} = E\left[y_1(i)\right] = \gamma_{1_p}(i) \tag{A.43}$$

Using the above equations, we get the following

$$p_{ij_{p+1}} = \frac{\sum_{t=1}^{T} \alpha_{t_p}(i)\, p_{ij_p} b_{j_p}(x_{t+1}) \beta_{t+1_p}(j)}{\sum_{t=1}^{T} \alpha_{t_p}(i) \beta_{t_p}(i)} \tag{A.44}$$

$$b_{j_{p+1}}(k) = \frac{\sum_{\substack{t \\ \forall x_t = v_k}} \alpha_{t_p}(j) \beta_{t_p}(j)}{\sum_{t=1}^{T} \alpha_{t_p}(j) \beta_{t_p}(j)} \tag{A.45}$$

Vita

Ihsan Ali Akbar was born in Karachi on March 16, 1978. He received his Bachelor of Engineering (B.E.) degree in Electrical Engineering from NED University of Engineering and Technology, Karachi, Pakistan in June 1999. He was enrolled as an M.S. Student at Virginia Polytechnic Institute and State University during Spring 2000, and later joined the Mobile and Portable Radio Research Group (MPRG) of Virginia Tech as a research assistant during the summer of 2000. From summer 2000 to December 2001, he remained actively involved in wireless channel modeling using discrete channel modeling techniques, and a major portion of his research involved the study of error statistics for third generation wireless channels, where he has published several technical reports in this area. He later joined the Ph.D. program of Virginia Tech in Spring 2003 and is pursuing a Ph.D. in electrical engineering. His research interests are in the areas of wireless communications including discrete and continuous channel models, digital and statistical signal processing, random processes, channel coding, array processing, and wavelets.