Adaptive Error Control for Wireless Multimedia

A Thesis
Presented to
The Academic Faculty

by

Andreas G. Yankopolus

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

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

Adaptive Error Control for Wireless Multimedia

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

I would like to thank Dr. Wicker and Dr. Copeland for their invaluable assistance, support, and encouragement through the Ph.D. program. Dr. Akyildiz, Dr. McLaughlin, Dr. Barry, and Dr. Toh were also always available with advice and assistance. I would also like to thank my wife Jennifer for tolerating the hours that writing this dissertation entailed and even marrying me in the midst of it.

ABBREVIATIONS

ABR	Available Bit Rate
ARQ	Automatic Repeat Request
ATM	Asynchronous Transfer Mode
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BMA	Berlekamp Massey Algorithm
BOP	Binary Operation
BPSK	Binary Phase Shift Keying
CBR	Constant Bit Rate
CSI	Channel State Information
FEC	Forward Error Control
FIR	Finite Impulse Response
GF	Galois Field
HMM	Hidden Markov Model
IP	Internet Protocol
ISI	Inter-Symbol Interference
LOS	Line Of Sight
MANET	Mobile Ad-hoc NETwork
MLSE	Maximum Likelihood Sequence Estimation
PH	Pseudo Harmonic
SNR	Signal to Noise Ratio
TCP	Transmission Control Protocol
TLA	Three Letter Acronym
UBR	Unspecified Bit Rate
UDP	User Datagram Protocol
QoS	Quality of Service

VBR	Variable Bit Rate

NOTATION

Table 1: Notation

symbol	example	meaning
number	1	scalar
bold number	$\mathbf{1}$	vector
lowercase letter	a	scalar
bold lowercase letter	$\mathbf{a}$	vector
single subscripted lowercase letter	a_i	vector element
bold uppercase letter	$\mathbf{A}$	matrix
double subscripted lowercase letter	$a_{i,j}$	matrix element (row, column)
colon	$\mathbf{a}_{i,:}$	vector element from a matrix
bold greek letter	$\boldsymbol{\alpha}$	vector or matrix
subscripted greek letter	α_i	vector or matrix element
parenthesis	$\mathbf{A}(\mathrm{n})$	time index
parenthesis	$\mathbf{A}(\mathrm{n}, \theta)$	time and parameter index

SUMMARY

Future wireless networks will be required to support multimedia traffic in addition to traditional best-effort network services. Supporting multimedia traffic on wired networks presents a large number of design problems, particularly for networks that run connectionless data transport protocols such as the TCP/IP protocol suite. These problems are magnified for wireless links, as the quality of such links varies widely and uncontrollably.

This dissertation presents new tools developed for the design and realization of wireless networks including, for the first time, analytical channel models for predicting the efficacy of error control codes, interleaving schemes, and signalling protocols, and several novel algorithms for matching and adapting system parameters (such as error control and frame length) to time-varying channels and Quality of Service (QoS) requirements.

CHAPTER I

INTRODUCTION

The use of wireless networks and multimedia applications continues to grow at a rapid pace. Such applications require that the wireless network deliver agreed upon Quality of Service (QoS) guarantees. Meeting such guarantees poses many problems due to the time-varying nature of wireless links and the differing QoS requirements of various multimedia applications.

This dissertation investigates several key problems that the designers of a wireless network will face:

- **Channel modeling**: How should one model wireless links during the network design process?
- **Error control**: How should one pick an appropriate error control code?
- **Adaptation**: How should the network adapt to varying channel conditions and different QoS requirements?

To these ends, this dissertation proposes a hidden Markov model (HMM) channel model along with a HMM design procedure to address the first problem, a method of predicting the performance of error control codes under expected channel conditions using the channel HMM to address the second problem, and a comparison of the decoding complexity of various error control codes and example adaptation scheme to address the third problem.

The models serve as valuable tools for quickly calculating many link performance figures such as packet error rate and throughput. The comparison of error control code decoding complexities allows the designer to balance performance with hardware requirements. The example "dual average" adaptation scheme offers a suggestion on varying the error control scheme based on measured channel conditions.

1.1 Motivation

The number of subscriptions to digital wireless services (e.g., cellular telephony, paging, broadcast television) and users of wireless networks (e.g., 802.11 LANs such as Orinoco's Wavelan and Cisco's Aironet products) has been growing at an incredible rate, as have the number of users accessing the Internet. At the same time, companies and researchers are introducing and experimenting with Internet multimedia applications (e.g., audio and video multicast, long distance telephony, distance learning, and video conferencing) that require the transmission of time sensitive audio and video data.

Service providers are now offering wireless Internet connectivity for stationary computer installations (e.g., wireless local loop service) laptop computers (e.g., Metricom and Motient) and the latest generation of PDAs (e.g, AT&T/Palm and Omnisky). Given the convenience and ease of installing such services in comparison to wired networks, one can only expect demand to increase. One can also expect the continued extension of multimedia services to mobile environments.

Quality of Service Multimedia applications employ different traffic types that have varying Quality of Service (QoS) requirements. For example, video and audio streams can typically tolerate packet losses fairly well but demand low end-to-end delays – in many cases it is better to drop the packet than deliver it late. Conversely, data transfer applications possess little sensitivity to end-to-end delay but tolerate packet losses quite poorly.

Meeting such QoS requirements when operating across a wireless channel poses numerous problems, as one has little control over the quality of a mobile communication channel, the condition of which can fluctuate wildly (e.g. fades of 30 dB). Hence, solutions designed for a worst case scenario will waste both bandwidth and power and systems designed for average conditions will stall when the channel worsens while still wasting resources during periods of good conditions.

This brings us to the need for a wireless communications system that will adapt to varying channel conditions and QoS requirements. For the purposes of this study, adaptations will be limited to error control coding (which includes interleaving, since an interleaver can be viewed as a rate one code) and packet length. In order to be useful and effective, the adaptation scheme should satisfy certain requirements:

- **Service guarantees**: The system should either satisfy the negotiated QoS requirements or inform the application that they cannot be met.
- **Transparency**: Adaptation should occur without intervention from the user.
- **Energy efficiency**: The adaptations should minimize power consumption in the case of battery powered devices.
- **Compactness**: The adaptation algorithm itself should be simple and efficient enough to operate in high-speed networks and on low power mobile hardware.

Multimedia applications operate by digitizing video and/or audio signals, arranging the bits into packets, and sending the packets to the receiver via the network. These packets will incur varying degrees of delay as they traverse the network, with the amount of delay depending on disparate factors (e.g., congestion, physical distances, switching speed, and serialization, which poses a particular problem across low-speed links.[1] The wireless link can introduce additional delays because of the error control system in use and cause errors due to the difficulty of communicating through a wireless channel.

Packets that arrive late or with errors are considered lost in the case of real-time multimedia applications. While these applications can tolerate a certain degree of packet loss, such losses will degrade the quality of the reconstructed audio and/or video signal.

Packets that arrive late will not cause problems for time insensitive applications, but those arriving with errors will require retransmission, thereby adding further load to the network.

The support of multimedia traffic in wireless networks poses many interesting and challenging design problems, as these applications require QoS guarantees, including:

[1]Serialization delay is often the dominant contributor to end-to-end delay in multihop low-speed networks. For example, 1500 byte packets (the maximum Ethernet payload size) take .1875 seconds to transmit across a 64 kbps data link, while 576 byte packets (a common size for TCP/IP implementations that do not use path MTU discovery) take 0.072 seconds. These delays can quickly add up across multiple links. Such low speeds are very common in DoD tactical radios: the SINCGARS radio, for example, operates at 1200, 2400, 4800, or 9600 bps in packet data mode.

- **End-to-end delay**: This is the maximum time it takes a packet to traverse the network from source to sink. It includes propagation delays, queuing delays, retransmissions, serialization delay, and processing times due to error control and interleaving.
- **Delay-jitter**: This is the the greatest difference in end-to-end delays suffered by any two packets.
- **Throughput**: This is the flow rate in bits per second seen by the application.
- **Error Rate**: This is the ratio of erroneous to total bits in the data stream. One is often more interested in the packet error rate, which is the ratio of packets containing one or more errors to the total number of packets.

Many of these design problems stem from the requirement to provide QoS guarantees in the time-varying mobile environment without wasting precious bandwidth and battery power.

The manner in which these requirements are agreed upon by the application and network depends on the underlying network traffic protocol. In connectionless networks (e.g., networks based on Internet Protocol (IP)), an additional service protocol (e.g., differentiated services (diffserv)) carries the QoS information inside the data packets. Or, an out-of-band signaling protocol allows the switches handling each flow to communicate QoS information (e.g., the Resource reSerVation Protocol (RSVP)). In connection oriented networks (e.g., Asynchronous Transfer Mode (ATM)), QoS negotiation takes place during the connection admission control phase of the circuit setup and is signaled by some other mechanism (e.g., Private Network to Network Interface (PNNI) in ATM networks).

In the case of PNNI, RSVP, or some other signaling protocol that reserves bandwidth in advance, the network will inform the application whether or not it can meet its QoS requirements. If not, the application may choose to wait until the network clears up or request lesser QoS requirements. If it can, the network will dedicate the resources necessary to maintain the QoS requirements for the life of the connection, with failure only occurring in certain overload conditions.

Error Control The addition of a wireless link complicates the equation, as the mobile radio channel is particularly hostile to communication. The characteristics of such channels can fluctuate widely and unpredictably. While wired links provide a low-noise environment for communication, wireless links suffer from:

- **Path loss**: The received signal strength decays with the square of the path length when radio waves propagate through free space. In mobile or indoor environments, the free space propagation model breaks down due to variations in antenna height and intervening obstacles. In such high-clutter terrain, the path loss exponent may be raised to four or higher.

- **Noise**: Thermal noise (produced internally by all electronic devices), man-made noise (e.g., automobile ignition systems and hairdryers) and natural noise (e.g., lightning) combine to corrupt the received waveform.

- **Co-channel interference**: The reuse of frequencies in cellular systems and wireless networks leads to co-channel interference, where the signal from other users interferes with the desired signal. Such interference has particular importance in code division multiple access (CDMA) networks, as all of the users transmit in the same frequency range and employ the same spreading codes to make each other appear as noise.

- **Fading**: Radio signals can reflect off and diffract around obstacles and arrive at the receiver with different phase angles. The multiple signals will interfere constructively and destructively, resulting in rapid fluctuations in the received signal strength. Fading is typically classified as either flat or frequency selective:

  - **flat fading**: This is the case where the inverse of the channel bandwidth exceeds the spread in propagation delays. Consequently, all of the frequency components in the waveform are affected equally and the received signal is relatively undistorted.

  - **frequency-selective fading**: This is the case where the spread in propagation delays exceeds the inverse of the channel bandwidth. As a result, the various frequency components comprising the waveform arrive at the receiver with varying amplitude and phase shifts. This causes significant distortion in the received signal.

- **Shadowing**: Shadowing refers to longer term variations in received signal strength caused by movement with respect to terrain features (e.g., entering a tunnel or parking garage). The received signal envelope due to shadowing often has a log-normal distribution.

These mechanisms will cause errors in the recovered bit stream, leading to packet losses, unless they are compensated for. Conventional digital cellular systems and wireless networks employ a fixed error control scheme if they use one at all. The receiver either drops erroneous packets or simply passes on the erroneous data. In either case, the radio link expects higher layer protocols to catch and handle the errors.

A data link control layer offering an arbitrarily low error rate would not necessarily provide the answer to this problem, as error control comes at a price in throughput, delay, and computational cost:

- **Delay**: An Automatic Retransmission reQuest (ARQ) scheme could introduce prohibitive delays due to the large number of retransmissions needed for successful reception of each packet. Interleaving the data and the use of long packets also introduces a queuing delay.

- **Throughput**: A given error code could require a code rate so low as to choke out the application.

- **Computational cost**: An error control code with sufficient power to provide an adequately low error rate might exceed the computational ability of the receiver side hardware.

As was alluded to earlier, multimedia applications are broadly classified into two categories:

- **Real-time**: These applications typically involve the interaction of two or more humans or a streaming audio and/or video feed. Examples include videoconferencing or streaming audio programming.

- **Non real-time**: These applications possess little sensitivity to delays. Examples include web browsing and file transfers.

Table 2 presents the QoS requirements of several real-time multimedia applications [2]. The Traffic Type column indicates the priority level of the traffic: deterministic, probabilistic, or best-effort.

Table 2: QoS Requirements for Various Real-Time Multimedia Applications

Channel Type	**Bandwidth**	**Jitter**	**Delay**	**Traffic Type**	**Error Rate**
Standard Video	25 Mbps	10 ms	250 ms	prob	10^{-3}
Slow Scan Video	1 Mbps	10 ms	250 ms	prob	10^{-2}
MPEG Video	1 Mbps	10 ms	250 ms	determ	10^{-9}
Voice Audio	64 Kbps	10 ms	250 ms	prob	10^{-1}
Hi Fi Audio	2 Mbps	5 ms	500 ms	determ	10^{-3}

Table 3 presents the QoS requirements of several non real-time multimedia applications.

Table 3: QoS Requirements for Various Non Real-Time Multimedia Applications

Channel Type	**Bandwidth**	**Jitter**	**Delay**	**Traffic Type**	**Error Rate**
Web Browsing	56 Kbps	500 ms	1 s	prob	10^{-7}
Remote Access	8 Kbps	250 ms	500 ms	prob	10^{-7}
Email	8 Kbps	N/A	N/A	best-effort	10^{-7}
File Transfer	8 Kbps	N/A	N/A	best-effort	10^{-7}

Communication links employ error control schemes to reduce the error rate in the recovered bit stream. Error control schemes operate by having the transmitter add controlled redundancy to the data stream in a manner that the transmitter and receiver have agreed upon beforehand. The receiver then uses the redundant data to check the validity of the received data stream and potentially to correct any errors that occurred in transmission. Generally speaking, the number of errors that the code can detect and/or correct increases with the quantity of redundant data. Hence, in a fixed data rate link, decreasing the error rate generally comes at a price in transmitted data rate.

The receiver and transmitter may utilize the controlled redundancy with one of three possible approaches:

- **Forward Error Correction (FEC)**: The receiver attempts to correct any errors that occur using the redundant information. The number of errors may exceed the code's error correcting capability.
- **Automatic Repeat Request (ARQ)**: The receiver uses the redundant information to verify the data stream. If it detects the presence of errors, it requests a retransmission of that data from the transmitter.
- **Hybrid ARQ**: This category includes schemes that utilize a combination of FEC and ARQ techniques. For example, the receiver may release a corrected version of the received data stream if detects the presence of less than a certain number of errors, but request a retransmission if the number of detected errors exceeds this limit. Hybrid ARQ systems may also combine multiple received packets in an effort to create a successfully decodable packet.

Each approach has its advantages and disadvantages. FEC systems may require the transmission of a large amount of redundant data to overcome poor channel conditions. However, they do offer a fixed delay and the lack of retransmissions makes them suitable for broadcast and multicast environments. ARQ systems require a reverse channel and introduce a variable delay into the network. They stand as the potentially least complex error control system and offer extremely low error rates for a given amount of redundancy. Hybrid ARQ systems combine the drawbacks of both FEC and ARQ systems and have the greatest potential complexity. In their favor, they can provide an excellent combination of delay, throughput, and low error rate.

The optimal error control strategy for multimedia applications will vary based the data stream's QoS requirements, the hardware available on either end of the link, and the channel conditions. As a rule of thumb, real-time applications generally require an FEC scheme and non real-time applications an ARQ scheme. A hybrid ARQ approach may also be an option depending on the circumstances.

Current wireless communication systems use a fixed error control code, if they use one at all. Wireless Ethernet (802.11b) 900 MHz systems use a simple error detection scheme and drop packets that arrive with errors, thus pushing the problem of reliability to a higher layer protocol (such as TCP). This approach can work quite well for non real-time applications, but is often unable to meet the QoS requirements of real-time applications.

Error control codes vary widely in their decoding complexity, a factor that the designer of a wireless network must take into account. The simple checksums used in 802.11b require only a linear feedback shift register to perform packet verification and can operate at very high data rates. The decoding of powerful Reed-Solomon codes capable of correcting large numbers of errors requires many more operations, while the computational requirements of turbo codes are great still. This can potentially pose a problem at high data rates or in battery powered hardware, such as the devices proposed for use in microsensor networks.

Channel Models The construction of hardware for field trials can require a considerable investment in time and funds. This motivates the development of practical and accurate mathematical models suitable for testing various systems on a computer.

Engineers rarely analyze modern wireless communications systems using traditional analytical methods. The reasons for this include the complexity of modern wireless communications systems and the numerous nonlinear and time-varying channel impediments which defy attempts to reduce the channel down to an Additive White Gaussian Noise (AWGN) model. Reducing the system to a tractable level typically involves making unacceptable simplifications, such as assuming that packets arrive at the receiver with a binomial distribution of errors.

Since these impediments are well known, one could conceivably write a computer simulation of the wireless communications system that would take the most importent of them into account. Such simulations, however, generally have an unacceptably long run time, precluding a thorough analysis or optimization of the system.

In an effort to strike a balance between the unrealistic assumptions necessary to apply traditional analytical tools and the excessive run times of comprehensive simulations, a number of hidden Markov Model (HMM) channel models were developed along with a general design procedure for calculating models for new channels. The use of HMMs to analyze communications systems dates back to Shannon's *A Mathematical Theory of Communication* where he used them to model a generalized noisy discrete channel[3]. This dissertation uses the states in HMM channel models created for this dissertation to represent the fading level of the channel. Representing the channel as a matrix allows for the calculation a wide variety of performance figures using advanced matrix algebra techniques.

1.2 Computer Analysis and Simulation

The performance of some communications systems, such as the BER of a deep space radio link, may be calculated using exact mathematical formulas. Such systems generally consist of an additive white Gaussian noise (AWGN) channel and a simple transmitter and receiver where these is no need to deal with Inter-Symbol Interference (ISI). The single number that results from the calculation gives an exact performance figure. This differs markedly from computer simulations, where the resulting number represents a random variable.

Even so, computers now play a major part in the analysis and simulation of communication systems. The reasons for this are not just the convenience of computer aided design tools, but incredible complexity of modern communications systems. One simply cannot model a cellular radio link as an AWGN channel due to the many channel impediments, such as multipath fading.

The engineer analyzing a communication system with a computer must take care to keep a close eye on possible sources of error. Computer calculations and simulations may exhibit errors originating from three possible sources:

- Numerical errors
- Parametric errors

- Model errors

Numerical errors rank as the easiest of the three to identify and control. Where possible, calculations were compared to those obtained by other researchers or calculated via other methods. As a result, these programs are expected to be free of programming errors that might lead to incorrect results.

Parametric errors occur because the parameters on which the calculations are based differ from those present in the real world, hence, simulated and measured results may differ due to parameter error. For example, Rician fading assumes that the receiver receives some combination of specular and diffuse signals from the transmitter where the ratio between the two is called the Rice Factor. Discrepancies between computer calculations and measured results caused by an incorrect Rice Factor in the calculations are parametric error. Such errors can be minimized through careful measurement. When developing a model, one should "jiggle" the various parameters in an attempt to understand how the model reacts to errors in parameter estimation.

Many of the results in this dissertation were obtained analytically using models in which the parameters were obtained from computer simulations. In these cases, the simulations were run numerous times and the mean and variance of the resulting parameter sets calculated. The variance was then used to determine how far to "jiggle" each parameter about the mean.

Model errors occur because the formulas used for investigating wireless channels are based on mathematical models that are necessarily simplifications of actual physical processes. For example, the power spectral density for the received signal envelope in a Rayleigh fading environment runs off to infinity at $f = \pm f_D$ (the doppler frequency). Naturally, this phenomenon is not observed in field measurements, as it is an artifact stemming from the assumption that the mobile station exists in a two dimensional universe in which the plane waves arrive from all directions with equal strength.

The models used in the dissertation research are based on ones that appear throughout the peer-reviewed literature on wireless networking. Models with the flexibility and accuracy necessary for this work. Where possible, results from the models in this dissertation were compared to simulation results and/or analytical calculations.

1.3 Contributions

The parallel deployment of fixed and mobile wireless broadband networks and QoS-sensitive applications raises the difficult question of meeting QoS guarantees in the unpredictable conditions of a wireless network. Users are not likely to accept lower QoS for their multimedia applications simply because they are not connected to a wired network. This situation suggests numerous research opportunities. The research summarized in this dissertation focussed on the problem of choosing error control codes to meet specific QoS requirements across fading channels. It was necessary to develop several channel modeling tools while investigating this problem, including a waveform-level GSM simulation, Monte Carlo models, and hidden Markov models.

This dissertation aims to develop useful tools for modeling wireless channels and choosing error control codes to meet QoS requirements. To that end, this dissertation makes the following contributions:

- Presentation of a detailed discussion of various simulation-based and analytical methods for assessing the performance of wireless channels. This analysis outlines the strengths and weaknesses of each method (chapter 3).
- Comparison of the suitability of Markov and hidden Markov models for estimating the number of erroneous symbols in a block of data transmitted across a fading channel and determination that hidden Markov models are suitable for most channels of interest but Markov models are generally not suitable (sections 4.4 and 4.5).
- Investigation of the accuracy of error distribution results from Markov and hidden Markov models created using different quantization schemes for the received signal (sections 4.4 and 4.5).
- Determination that minimum mean-squared error (MMSE) quantization of the Rayleigh envelope does not lead to accurate channel models (sections 4.4 and 4.5).

- Proposing and demonstrating a logarithmic quantization scheme well suited to modeling error occurrences in a Rayleigh fading channel (section 4.2).
- Presenting an in-depth presentation of the steps and utilities required to model a wireless channel using hidden Markov models (section C).
- Demonstrating the utility of hidden Markov models for assessing the performance of wireless channels in terms relevant to meeting QoS guarantees (section 4.6).
- Outlining a methodology for determining an "optimal" error control code given a hidden Markov model for the channel and a set of QoS constraints (section 4.7).
- DIscussion of techniques for adapting error control codes to varying channel conditions, including a novel "dual average" algorithm (section 4.7).

1.4 Outline

The remainder of this dissertation is organized as follows: Chapter 2 provides an overview of related work. Chapter 4 presents the important results from the calculations and simulations behind this dissertation. In particular, the complexity and performance of error control codes is discussed from the viewpoint of QoS guarantees. Finally, Chapter 5 summarizes the dissertation and suggests several avenues for future research. The appendices discuss matrix probabilities and the fitting of Markov models to communications channels.

CHAPTER II

RELATED WORK

This section discusses some of the prior work that has been done in the areas of adaptive error control and channel modeling using Markov models. The papers mentioned below are not presented as an exhaustive list, but as a subset of the papers read during the course of this research that were found particularly useful.

2.1 Adaptive Error Control Coding

Two early papers on adaptive error control were written by Nachum Shacham. The first paper [4] presents a high-level overview of techniques for dynamic parameter selection in time-varying channels. The second paper [5] discusses an adaptive hybrid ARQ error control architecture with two codes of different rates. The code in use at any time is chosen based on the error history of the channel. The use of two codes allows the long-term throughput across the channel to exceed the throughput when using only one of the two codes the entire time.

Rice and Wicker [6] presented an adaptive error control scheme for time-varying channels that uses a combination of FEC and ARQ techniques. The authors utilized a Markov channel model and examined the throughput of various code combinations and presented a method for choosing optimal frame lengths.

Yajnik, Sienicki and Agrawal [7] examined the relationship between error coding and connection setup delay for two state Markov model channels. The authors also presented a technique for choosing an encoding scheme based on a channel BER and delay bound.

Schuler [8] examined the optimization of FEC and ARQ algorithms to meet ATM QoS guarantees. The author examined adaptation to the channel error rate in light of the required QoS, which involved target BERs for FEC codes, delay for ARQ codes, and both parameters simultaneously for hybrid systems.

Gomez, Campbell and Morikawa [9] presented a framework for QoS prediction, compensation, and adaptation in wireless networks. The paper discussed adaptations at both the packet and application level to account for both short and long term adaptations to changing conditions. A two-state markov channel was used to model the arrival of correct and erroneous packets.

Elaoud and Ramanathan [10] presented an adaptive FEC scheme that selected one of several codes. In addition, the scheme would defer packet transmissions during extremely poor channel conditions, such as during deep shadowing. The authors showed their system, via simulation, to outperform a single FEC code system.

Lettieri and Srivastava [11] examined the transmission of data across a wireless link from the standpoint of efficiency and battery drain. They focused on varying the length of the MAC layer frame and presented comparisons of goodput and energy efficiency versus channel BER for varying frame lengths. The analysis demonstrated the utility of decreasing the frame length as channel conditions deteriorated and vice versa.

Chen et al. [12] modeled the wireless channel with a Markov model fitted to the received signal strength. The adaptation scheme consisted of three error control codes, each associated with the state of the system. The system switched between the stated based on the the received signal strength.

Chien et al. [13] investigated the adaptation of frame length, error control coding, and processing gain with the goal of minimizing expended battery energy. The paper presents an adaptive radio with "control knobs" that may be used to dynamically these parameters. Through analysis and simulation, the authors show that their radio utilizes significantly less power and provides better performance than a radio tuned for typical channel conditions.

Akyildiz et al. [14] proposed an adaptive FEC scheme "Yurie FEC" for wireless ATM networks. The scheme utilized Reed-Solomon codes and allowed for TCP operation at bit error rates of up to 10^{-3}. Another novel AAL "AAL X" [15] [16] added an adaptive frame length mechanism and status messaging that allowed for the differentiates between packets lost due to error and packets lost due to congestion.

2.2 Channel Modeling

Claude Shannon utilized Markov models to represent both sources and channels in his landmark paper [3] that established the field of information theory. Gilbert's paper [17] introduced the use of a Markov error source. The model presented in this paper has two states: a good state in which errors never occur and a bad state where errors occur with probability $1 - h$.

Frichtman published a more complex model [18] which utilizes multiple good states, in which errors never occur, and one bad state, in which errors occur with probability one. Transitions are not allowed between two good states; a good state may only transition to itself or the bad state.

William Turin [19] a described a novel solution to a problem that can arise when using the Baum-Welsch Algorithm (BWA) to estimate the HMM for an error process when the error events are widely spread apart. The BWA provides an efficient method for calculating the parameters of a HMM given an observed (or simulated) data set of events. The algorithm grew out of research in speech recognition, but quickly found applications across electrical engineering. In addition to its efficiency, the BWA is robust in that it will always converge to a local maximum of the parameter likelihood function. This approach works well when the events in the data set occur fairly frequently, but the computational and memory requirements become unreasonable when the intervals between events grow large.

Swarts and Ferreira [20] measured the actual channel statistics of a VHF mobile propagation channel to create a Frichtman model of the channel. The model provided an accurate estimate of bit error rates, but did not fare as well in predicting the distribution of errors within a burst.

Zorzi, Rao, and Milstein [21] assessed the accuracy of a simple first-order Markov model for a Rayleigh fading channel. The analysis found that even this simple model can provide a fairly accurate approximation of the success/failure probabilities of received data blocks.

Wang and Moyeri [22] analyzed a Markov Model in which each state was associated with a binary symmetric channel. This was accomplished by partitioning the received SNR into a finite number of intervals, where each interval was associated with a particular state.

The authors argued that their model provided an accurate description of a Rayleigh fading channel based on the calculated state transition probabilities.

Another variation on the Baum-Welsch Algorithm appeared in a paper by Sivaprakasam and Shanmugan [23]. The authors found a technique for relaxing the constraints of the technique presented in [19]. This paper also encouraged the use of models with multiple bad states, instead of the single state suggested in [18] and generally used by researchers.

Sajadieh, Kschischang, and Leon-Garcia [24] investigated methods to model the state holding times for each quantization interval of a birth-death Markov model of a Rayleigh fading process. The authors derived the state transition probabilities from the statistics of the Rayleigh fading process. Through the use of two states for each quantization interval, the authors approximated the gamma-type distribution of the dwell time of the Rayleigh fading process within each quantization interval.

Turin and van Nobelen [1] examined the use of hidden Markov models (HMMs) for modeling a fading process and showed the utility of the numerous closed-form solutions available for HMMs. The authors evaluated several approaches for matching the parameters of the HMM to observed data and illustrated the mathematical techniques for calculating the state dwell time distribution, fade duration, and level crossing distribution.

Babich and Lombardi [25] utilized an information theoretic approach to examine the accuracy of two and three level Markov models for modeling a fading process. Theis analysis suggested that this model may fully characterize channel behavior, but the authors suggested that such simple models are inadequate for approximating some fade rates.

Zorzi, Chockalingam, and Rao [26] analyzed the performance of several different versions of TCP using a Markov channel model. The model described a simple packet transmission success / failure process and was combined with the operating procedures behind the TCP versions to calculate the throughput for different fade rates and SNRs.

Tan and Beaulieu [27] examined the validity of analyzing fading channels using first order Markov models using an autocorrelation based approach. The authors concluded that a first order model can be useful if only a few consecutive channel samples are needed, but is unsuited for applications requiring numerous consecutive samples.

CHAPTER III

PRIOR WORK

3.1 GSM Simulation

The widespread interest in GSM[1] spurred the development of a flexible waveform-level GSM simulator. This simulator was intended as a testbed for investigating a variety of error control codes and techniques.

3.1.1 Introduction

The work for this dissertation began with the writing of a computer simulation that would serve as a testbed for new ideas. The intent was to use this simulator as a foundation on which to test a variety of error control techniques in an effort to recognize patterns and generalize broad results about wireless network QoS.

The intent was to adjust the codes and interleaving to meet a variety of QoS requirements given different current channel conditions. To this end, the simulator was primarily run with an inner convolutional code (specified by the GSM standard) and an outer Reed-Solomon code, the two of which were separated by an interleaver. The output from the inner convolutional code was further interleaved before transmission across the simulated wireless channel.

The testbed intially focussed on GSM,a widely-deployed second-generation cellular system at the time. Although the simulator specifically addressed GSM, it was written in a modular fashion so that the functional elements could be reused to simulate other digital communications systems.

The simulator included a variety of modules:

- GMSK modulator and demodulator

[1]Originally Groupe Spéciale Mobile, but since changed to Global System for Mobile Communications. This did not change the acronym.

- MLSE channel equalizer
- SOVA MLSE channel equalizer
- Convolutional encoder
- Viterbi decoder
- SOVA Viterbi decoder
- Reed-Solomon encoder and BMA decoder
- Rayleigh fading generator
- τ-spaced tapped delay line (for COST 207 [28] multipath fading)
- Interleaver and deinterleaver (GSM interleaver and block interleaver)

The simulator also has support functions that allowed it to perform a variety of user-configurable tasks, such as looping through a list of SNRs, simulating a given subscriber velocity, and transmitting a user-defined volume of data.

3.1.2 Results

The following paragraphs show some example results obtained with this simulator. The example plots for this system with the HTx6 channel (hilly terrain, six rays) and a vehicle speed of 100 km/h. The simulations shown in this section were run using either a (240,230) or a (240,220) Reed-Solomon outer code (a (255,245) or (255,235) code shortened by 15 symbols) and the standard GSM convolution inner code. The length of the Reed-Solomon codes was chosen to mesh well with the TCH/F9.6 block length of 60 bits. Thirty-two such blocks (1920 bits) are used to carry each Reed-Solomon codeword.

The Reed-Solomon codewords are passed through a block interleaver (which operates at the symbol level) with an edge length of 25 symbols before entering the GSM convolutional encoder. The output from the convolutional decoder is further interleaved before the bits are arranged onto GSM bursts. The resulting bits are distributed over eight time slots on eight different carriers to give a transmitted bit rate of 73,600 bps ($8 \cdot 9600$bps). The (240,230) code has a rate of .9584 which lowers the effective transmitted bit rate to 73,600 bps while the (240,220) code, with a rate of 0.9167, lowers the bit rate to 70,400 bps.

The probability of the decoder catching an erroneous codeword is 0.99 for the (240,230) code and 0.9999997 for the (240,220) code using (4.39). Hence, one would expect approximately one out of every hundred erroneous codewords to pass by the (240,230) decoder but only about three out of every ten million to escape the (240,220) decoder. Simulation results bore out these values: the (240,230) decoder occasionally approved an incorrect codeword while the (240,220) flagged every incorrect codeword.

The simulation results shown in Figures 1, 2, 3, and 4 involved the transmission of 245,760 bits at Eb/N0s of 0dB to 30dB in increments of 2dB.

Figure 1 shows the BER of the received bitstream at the output of the channel equalizer and at the output of the Viterbi decoder for both hard and soft (SOVA) MLSE equalization. One can clearly see that the convolutional code, despite its simplicity, provides a tremendous coding gain. Furthermore, SOVA MLSE equalization provides an additional 5 to 10dB in coding gain!

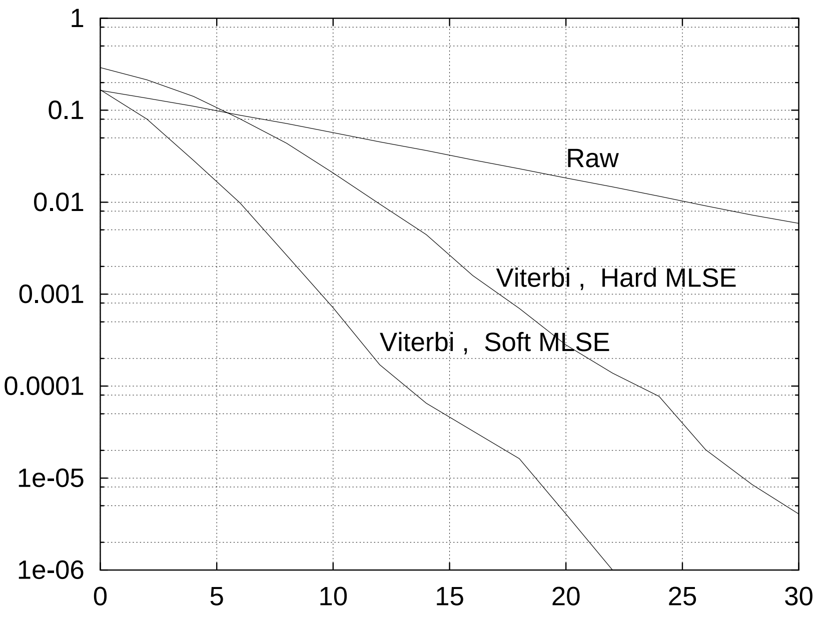

Figure 1: GSMsim BER Plot

Figure 2 plots the BER of the bitstream leaving the Reed-Solomon decoder. Received vectors which resulted in a decoder failure had their parity bits removed and were copied to the output unchanged for BER computation purposes. One can see that soft decision decoding improved the performance by approximately 6dB.

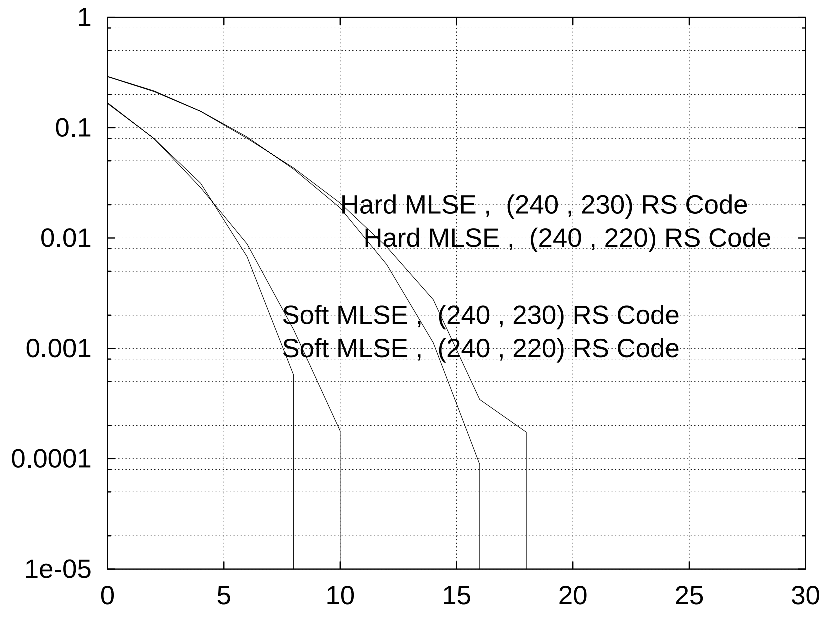

Figure 2: BER, All RS Codewords

Figure 3 gives the fraction of packets which were discarded by the receiver due to a Reed-Solomon decoder failure. From this figure, one can clearly see the improvement wrought by soft decision MLSE equalization. At an Eb/N0 of 12.5dB, for example, the hard decision systems discarded virtually every packet while the soft decision systems discarded only a fraction of the packets.

Figure 4 compares the bit rate of these two systems. Over most of the range, the (240,220) code provided a significantly higher throughput even though it has a lower rate (.9167 vs. .9583). Above an Eb/N0 of 10 (hard MLSE) or 16 (soft MLSE) the higher rate code won out. Again, soft decision MLSE equalization provides a gain of approximately 6dB. This figure emphasizes the importance of switching codes when possible.

3.1.3 Summary

Running simulations such as those outlined above provided a great deal insight into various aspects of wireless system performance. However, altering the simulation required considerable coding effort: analyzing a different interleaving, coding or modulation scheme required

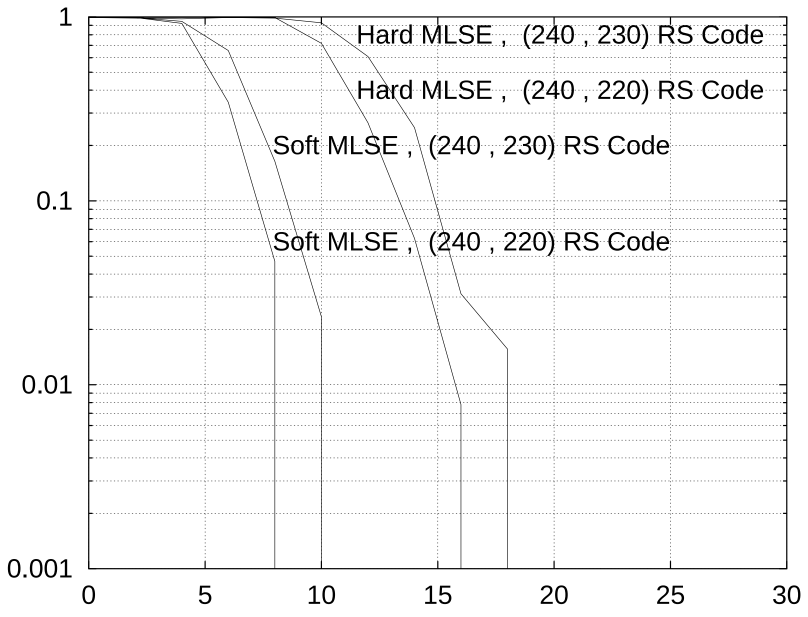

Figure 3: Packet Rejection Rate

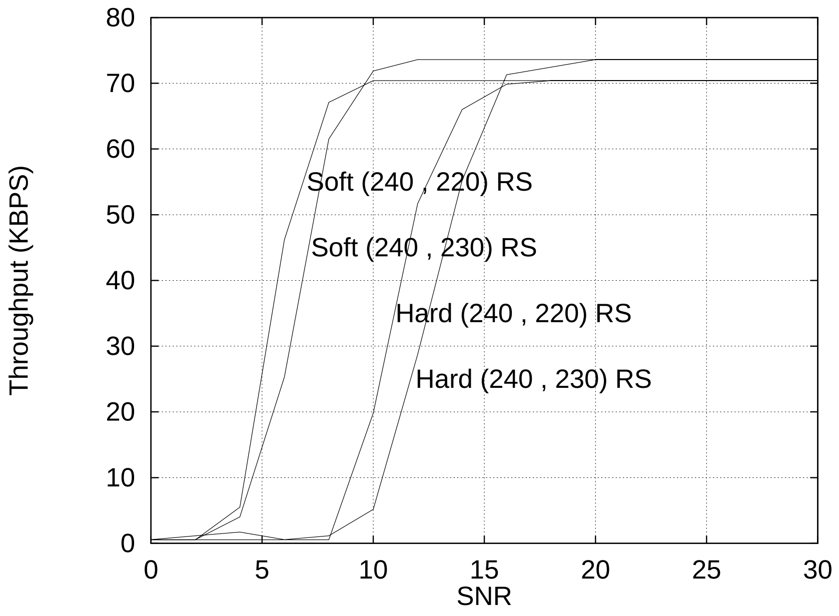

Figure 4: Throughput

the writing of code that simulated that scheme on a very detailed level. The use of powerful error correcting codes turned errors into very rare events. This necessitated excruciatingly long simulation execution times even though the code in the simulator had been heavily optimized.

Furthermore, much as it is impossible to “prove” something using analogies, it is also impossible (except for a few special cases) to construct a “proof” based on Monte Carlo simulations. Given the benefit of hindsight, the writing of this simulator and the execution of numerous simulations provided a useful intuitive feel for the effects of different coding, interleaving, equalizing and modulation schemes, but little hard data.

3.2 Analytical Methods

An effort was made to examine possible methods for analytically calculating the performance of different codes in a fading environment to overcome the limitations of computer simulations. The work of François Gagnon and David Haccoun, as outlined in their paper “Bounds on the Error Performance of Coding for Nonindependent Rician-Fading Channels” [29] served as a starting point. The interested reader is urged to read their paper in its entirety, as the discussion below merely touches on the points relevant to the ideas advanced in this paper.

3.2.1 Introduction

The procedure for calculating the performance of an error control code across a memoryless channel, such as the ubiquitous additive white Gaussian noise (AWGN) one, is fairly straightforward [30]. The degree of accuracy attainable in the analysis depends entirely on knowledge of the code's weight distribution[2]. An exact residual bit error rate can be completed if the complete weight enumerator is known. Otherwise, the computations are limited to upper bounds on performance, although these upper bounds can be very tight.

For more complex channels, such as one with correlated fading, the analysis becomes both more complex and computationally intensive. The degree to which each codeword

[2]See Appendix A for a brief description of weight and weight distribution

contributes to the residual error rate depends on the position of each code symbol in the codeword as a result of the correlation of the fading process. Since the positions are unique to each codeword, one cannot calculate the contribution of every codeword of a given weight at once: every codeword must be checked individually for its contribution. This can pose a seemingly insurmountable computational burden, since, for some codes, the cardinality of the codespace is too large for an exhaustive search.

Despite the influence of each codeword's fine structure, its weight still plays the dominant role in determining its contribution to the undetected error probability. Even though the high-weight codewords tend to contribute less per codeword to the error rate, there are more of them. Hence, the bulk of the contribution to the error rate could come from middle weight codewords.

One way to lessen the computational burden is immediately obvious: compute the error sum by starting with the lowest weight codewords and add codewords of greater weight until their contribution is no longer significant. While good in theory, this method poses the practical problem of requiring a priori knowledge of all of the codewords. Additionally, the number of codewords of each higher weight grows extremely rapidly for some codes.

This new algorithm quickly arrives at an accurate estimate of the undetected error probability regardless of the cardinality of the codespace and requires only knowledge of the error control code's weight enumerator and systematic generator matrix. It will work with a nonsystematic generator matrix, but with a much slower convergence rate.

3.2.2 Background

For a transmitted sinusoidal waveform in a Rician fading environment, the normalized received signal may be written as [31]:

$$r(t) = \alpha\cos(\omega_c t) + S_c(t)\mathrm{cos}(\omega_c t) + S_s(t)\mathrm{sin}(w_c t) + n(t)$$

where $S_c(t)$ and $S_s(t)$ are stationary zero mean Gaussian random processes and $n(t)$ is a white Gaussian noise process. If $S_c(t)$ and $S_s(t)$ have variance σ^2 then one can define the ratio of specular to diffuse energy as $\gamma = \frac{\alpha^2}{2\sigma^2}$.

For an arbitrary (n, k) linear block code, one can enumerate the set of all codewords C as $\{C_0, C_1, C_2, \cdots, C_{2^k-1}\}$ where C_0 is the all-zeros codeword and the information vector corresponding to C_j has weight B_j. The probability of receiving codeword C_j if C_0 was sent is then given by:

$$p_{C_j,C_0}(z) = \frac{e^{-\frac{1}{2}z\alpha^2 T_c \mathbf{B}(\sigma^2 T_c \mathbf{L} z + \mathbf{I})^{-1}\mathbf{B}'}}{|\sigma^2 T_c \mathbf{L} z + \mathbf{I}|} \tag{3.1}$$

where $\mathbf{B} = [1, 1, \cdots, 1]$, z is a paramater dependent on noise level and modulation scheme, $\mathbf{L}$ is the autocovariance matrix for C_j and T_c is the symbol duration. If the B_j nonzero entries of C_j are labeled $a_1, a_2, \cdots, a_{B_j}$ then the entries of autocovariance matrix $\mathbf{L}$ are written as:

$$\mathbf{L}_{ij} = \rho(|a_i - a_j|T_c)$$

where T_c is the coherence bandwidth of the channel and $\rho(\cdot)$ is the channel autocorrelation function (typically $e^{-2\pi F\tau}$, where F is the Doppler frequency in Hz). For example, $\mathbf{L}$ for the codeword [1001101] equals:

$$\begin{bmatrix} 1 & \rho(3T_c) & \rho(4T_c) & \rho(6T_c) \\ \rho(3T_c) & 1 & \rho(T_c) & \rho(3T_c) \\ \rho(4T_c) & \rho(1T_c) & 1 & \rho(2T_c) \\ \rho(6T_c) & \rho(3T_c) & \rho(2T_c) & 1 \end{bmatrix}$$

The probability of undetected bit error is given by:

$$P_b \leq \frac{1}{k} \sum_{j=1}^{2^k} B_j \mathbf{E}[P_e(C_j, C_0)] \tag{3.2}$$

where B_j is the weight of the information sequence associated with C_j, k is number of information bits and $\mathbf{E}[P_e(C_j, C_0)]$ is the expected value of the error probability for $\mathbf{C}_j$.

For maximal ratio combining with unquantized demodulation of PSK modulated signals, $\mathbf{E}[P_e(C_j, C_0)]$ is upper bounded as:

$$\mathbf{E}[P_e(C_j, C_0)] \leq \left. \frac{p_{C_j,C_0}(z)}{2} \right|_{z=\frac{1}{N_0}} \tag{3.3}$$

One can substitute the expression for p_{C_j,C_0} from (3.1) into (3.3) and use (3.2) to arrive at:

$$P_b(N_0) \leq \frac{1}{2k} \sum_{j=1}^{2^k} B_j \frac{e^{-\frac{1}{2N_0}\alpha^2 T_c \mathbf{B}(\sigma^2 T_c \mathbf{L}\frac{1}{N_0}+\mathbf{I})^{-1}\mathbf{B}'}}{|\sigma^2 T_c \mathbf{L}\frac{1}{N_0} + \mathbf{I}|} \tag{3.4}$$

which is the equation for summing together the contribution of every single codeword. The equation for the error contribution of a single codeword is found by removing the summation from (3.4). This gives the working expression:

$$P_b(C_j, N_0) \leq \frac{B_j\, e^{-\frac{1}{2N_0}\alpha^2 T_c \mathbf{B}(\sigma^2 T_c \mathbf{L}\frac{1}{N_0}+\mathbf{I})^{-1}\mathbf{B}'}}{2k|\sigma^2 T_c \mathbf{L}\frac{1}{N_0} + \mathbf{I}|} \tag{3.5}$$

(3.5) requires one to invert and take the determinant of a matrix with edge length equal to the codeword weight, a task that becomes increasingly computationally intensive as the edge length grows. For this reason, reducing the number of times (3.5) is extremely important, especially for high weight codewords.

3.2.3 Algorithm Description

This algorithm uses Monte Carlo techniques with importance sampling [32] to quickly arrive at an accurate estimate of the error probability. When using Monte Carlo techniques, it is critical that the sampled data points accurately represent the entire sample space. This algorithm accomplishes this goal by segregating the codewords in the codespace by weight and taking random codewords from each weight to estimate the error probability. The weights found to contribute most heavily to the error probability are sampled more thoroughly to improve the estimate.

The algorithm begins by creating a set of error bins. Each individual error bin holds the error contribution for one codeword weight (Figure 5). The algorithm then generates a given number (T_1) of random information vectors of each possible weight. These vectors are multiplied by the code's generator matrix to compute the associated codewords. The error contribution of each of these codewords is calculated individually using (3.5) and added to the respective error bin.

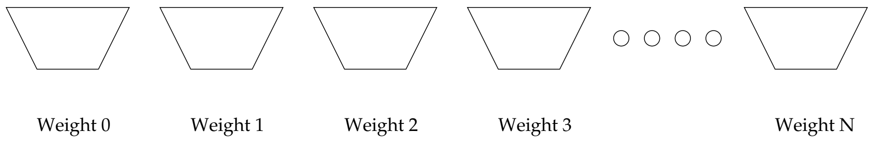

Figure 5: Error Bins

An estimate of the undetected error probability is calculated by scaling the error in each error bin by the percentage of codewords of that weight in the error bin:

$$\hat{P}_b = \sum_{j \in W} e_j \frac{A_j}{a_j} \tag{3.6}$$

where W is the set of codeword weights, e_j represents the total error in the weight j error bin, a_j the number of codewords contributing to e_j and A_j the total number of weight j codewords.

For example, consider the hypothetical code in table 4 that has codewords of weights 5, 7 and 9. After computing this initial error estimate, the algorithm determines the range of codeword weights which contributed most heavily to the error probability. This range is selected for further examination.

Table 4: Hypothetical code

weight	5	7	9	sum ($\hat{P}_b$)
sampled codewords (a_j)	25	20	10	
actual codewords (A_j)	500	1000	500	
fraction	1/20	1/50	1/50	
multiplier ($\frac{A_j}{a_j}$)	20	50	50	
error sum (e_j)	5e-4	6e-5	1e-5	
estimated contribution ($e_j \frac{A_j}{a_j}$)	1e-2	3e-3	5e-4	1.35e-2

Suppose the range runs from w_{min} to w_{max} for a systematic (n, k) block code. Although no straightforward method exists for creating a codeword of a given weight, a data vector of weight w will map to a codeword with weight from w to $w+(n-k)$. Hence, in the search for codewords with weights ranging from w_{min} to w_{max}, one should pick random data vectors with weights from $w_{min}-(n-k)$ to w_{max}.

Random data vectors of each weight are chosen without replacement for weights where the total number of vectors is fairly small. This avoids the possibility of double counting when the probability of selecting the same data vector twice is significant. The vectors are chosen with replacement for weights where the chance of double counting is small. The reason for this difference is that the memory and processor requirements for choosing without replacement increase rapidly with the number of vectors while the added precision decreases equally rapidly. The largest number of codewords sampled without replacement is T_2. Thus, if there are more than T_2 codewords of a given weight, that weight is sampled with replacement. Otherwise, it is sampled without replacement.

The error contributions for these new codewords are added to the error bins and a new estimated probability of error is calculated. If this new estimate lies within a given interval around the previous estimate, the algorithm halts. Otherwise, a new range of codeword weights to check is calculated and the algorithm repeats. This process continues until the error estimate converges or the codespace is sampled exhaustively.

The error estimate converged within a few iterations in every one of many tests. The standard deviation of the samples in each weight bin has also converged in every case. This offers additional evidence that this algorithm is choosing a representative sample of the codespace and, consequently, providing an accurate estimate.

3.2.4 Numerical Results

Several different codes were used to verify this technique of which a few had a codespace small enough that the error probabilities could be calculated using both an exhaustive search and the Monte Carlo algorithm. The double error correcting binary BCH (31,21) code is one of the codes that satisfied this criterion; the results are shown in Figure 6.

The autocorrelation function used for all of the calculations was $\rho(\tau)=e^{-2\pi F\tau}$ where F is the Doppler Frequency. Each iteration used $T_1=1000$ data vectors per weight, which were chosen without replacement for weights with less than $T_2 = 100,000$ associated codewords. After each iteration, the codeword weights contributing 99% of the sum were chosen for further sampling if the previous iteration altered the error sum by more than 2.5%. These values were chosen through empirical methods but seem to work well.

As a result of these Monte Carlo methods, the number of codewords used to calculate each data point dropped from 2.1×10^6 to about 6×10^4. A comparison of the results for these two methods appears in Figure 6 and the percent error for the Monte Carlo method in Figure 7.

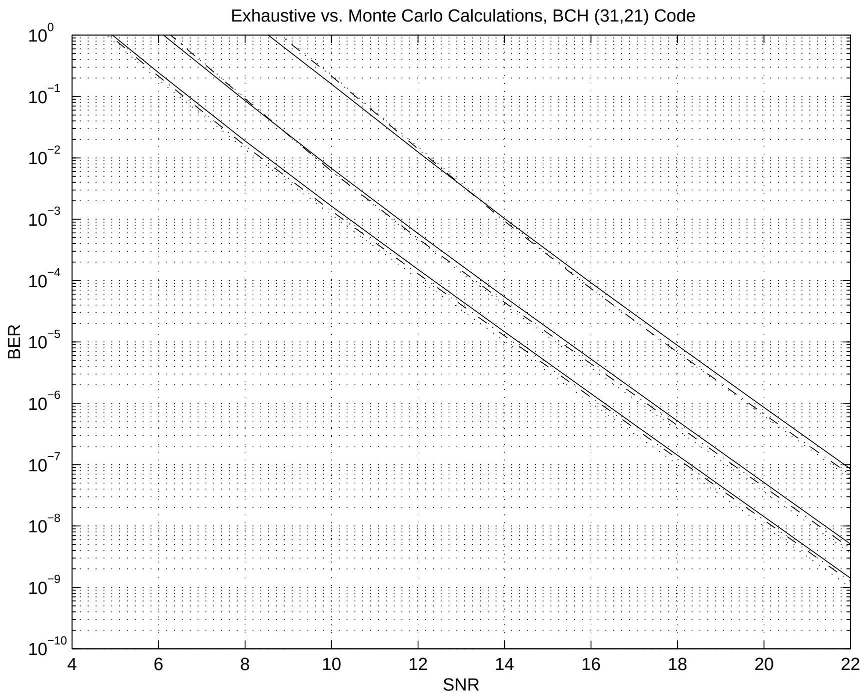

Figure 6: Exhaustive (solid line) and Monte Carlo calculations (dashed line) for $\gamma = 0$ and $FT_c = 0.1$ (bottom pair), 0.03 (middle pair) and 0.01 (top pair).

Reducing the number of data vectors per weight to $T_1 = 100$ and the threshold for sampling without replacement to $T_2 = 10000$ gave the BER curves in Figure 8 and the percent error curves in Figure 9. The total number of codewords utilized for each data point was about 6×10^3. Reducing the number of codewords used in the estimation did increase the estimation error, although the maximum error remained less than 35%. This translates to a fraction of a dB.

The additional iterations typically push the initial estimate up or down a few percent.

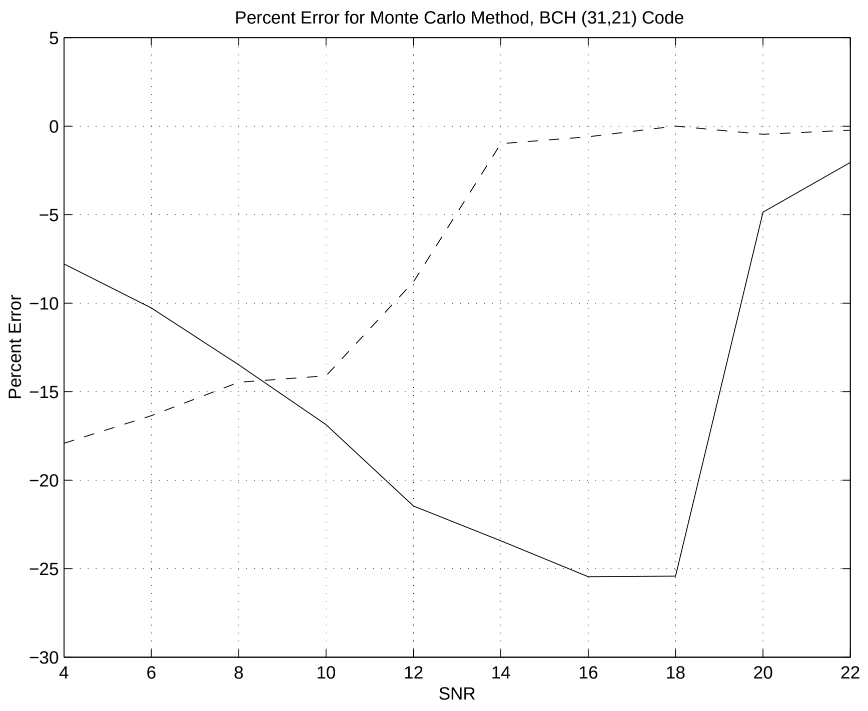

Figure 7: Percent error for Monte Carlo method, $FT_c = 0.01$ (solid) and 0.1 (dashed).

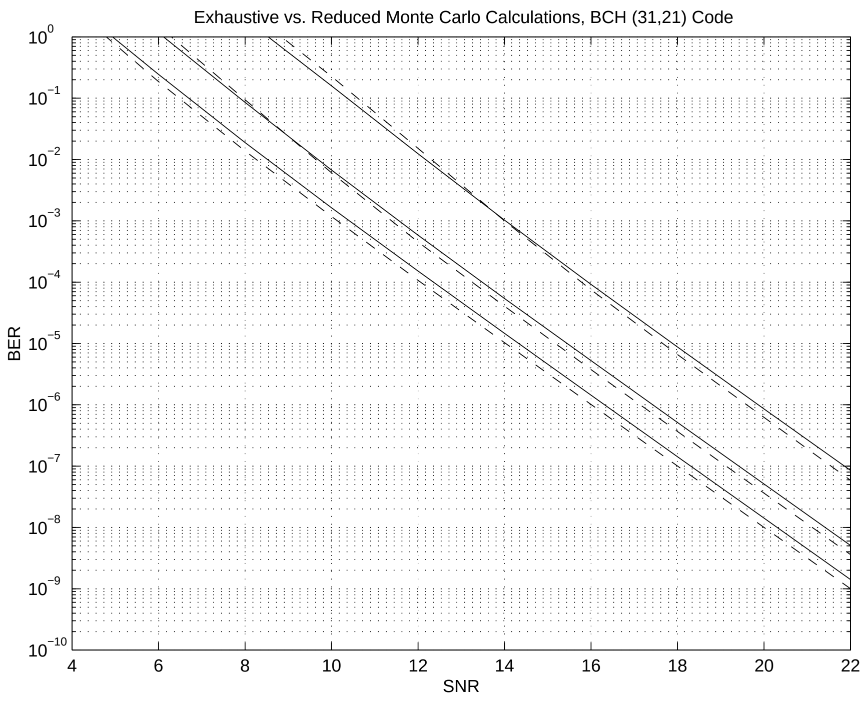

Figure 8: Exhaustive (solid line) and reduced Monte Carlo (dashed line) for $\gamma = 0$ and $FT_c = 0.1$ (bottom pair), 0.03 (middle pair) and 0.01 (top pair).

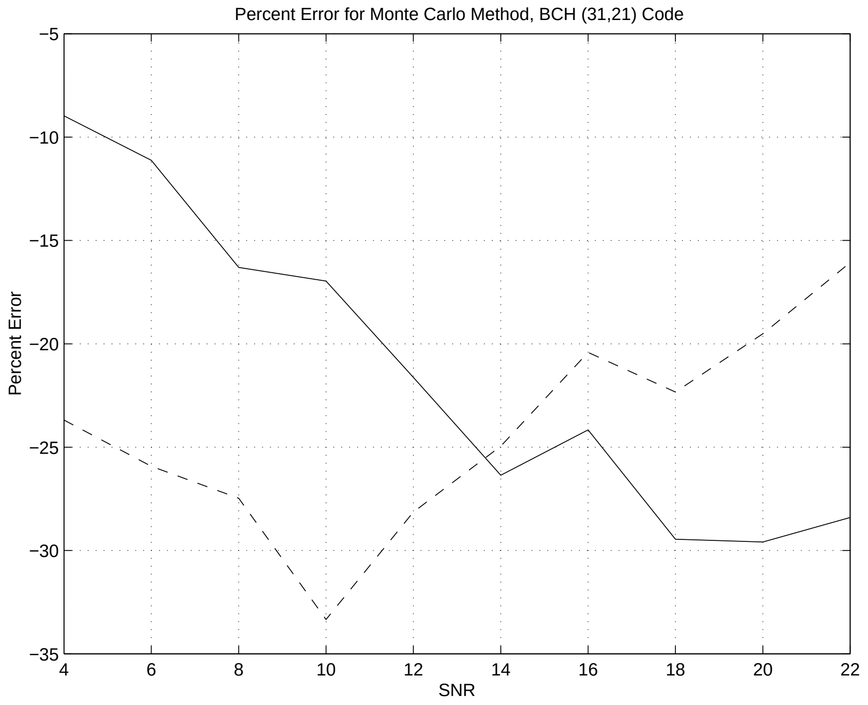

Figure 9: Percent error for Monte Carlo method, $FT_c = 0.01$ (solid) and 0.1 (dashed), reduced computations

Figures 10 and 11 present two typical cases.

It should be noted that the number of codewords checked by the Monte Carlo method increases linearly with the codespace's dimension. This stands in stark contrast to an exhaustive search, where the number of codewords checked increases at an exponential rate. Switching to a BCH (63,51) code gives the exhaustive method 2.6×10^{15} codewords to search through. The Monte Carlo method only looks at about 10^5 codewords given the thresholds of $T_1 = 1000$ and $T_2 = 100000$ and 10^4 for $T_1 = 100$ and $T_2 = 10000$. The BER curves for these two calculations appear in Figure 12.

3.2.5 Summary

This section presented a fast and accurate method of estimating the residual error rate of an error correcting code has been presented. The time savings allow for the analysis of large codes provided that the weight enumerator is known and a systematic generator matrix exists.

Unfortunately, this approach suffers from several problems that are not easily addressed.

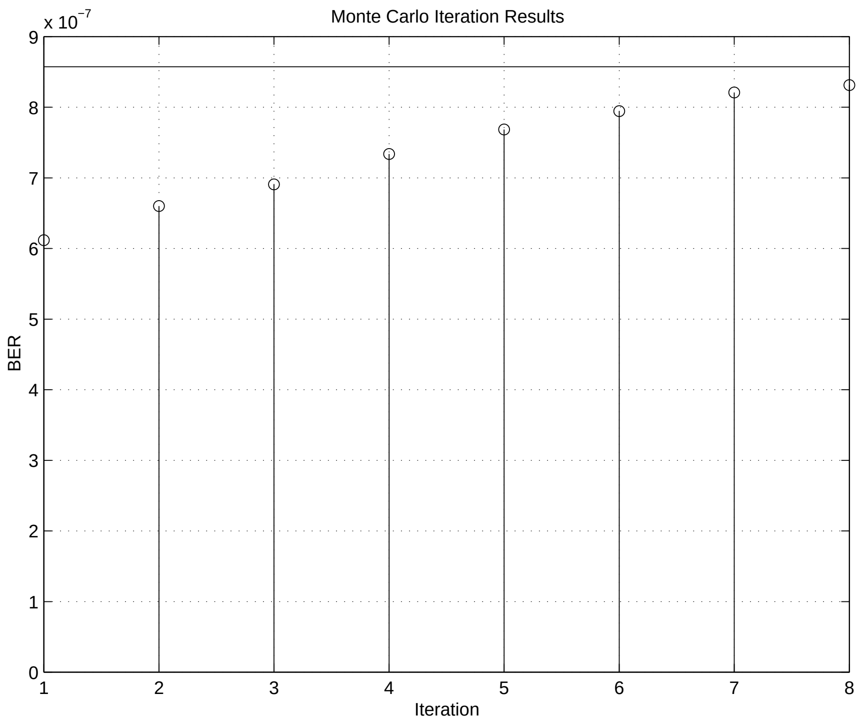

Figure 10: Monte Carlo convergence for $FT_c = 0.1$, $SNR = 20$

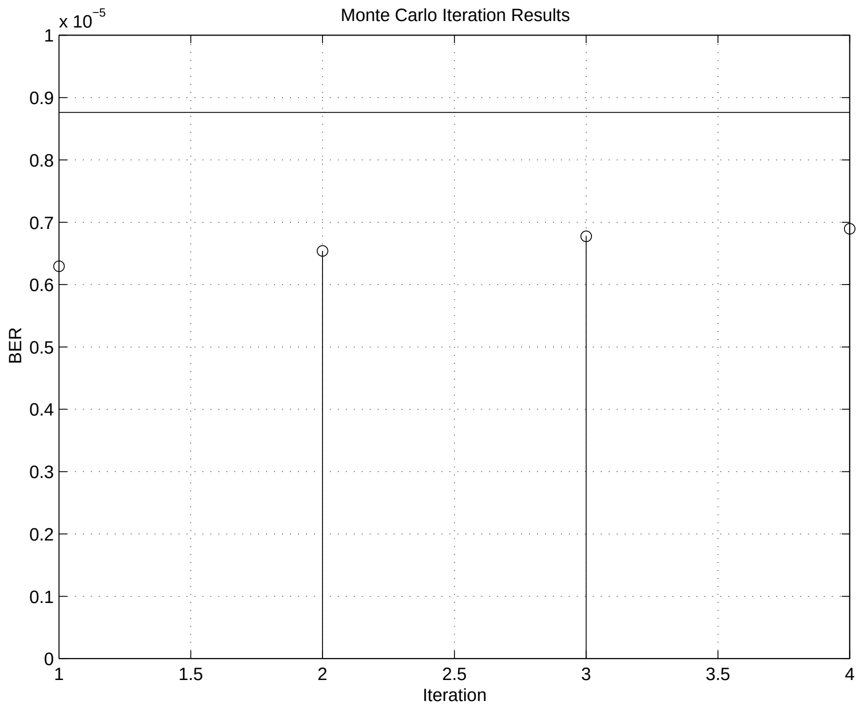

Figure 11: Monte Carlo convergence for $FT_c = 0.1$, $SNR = 18$

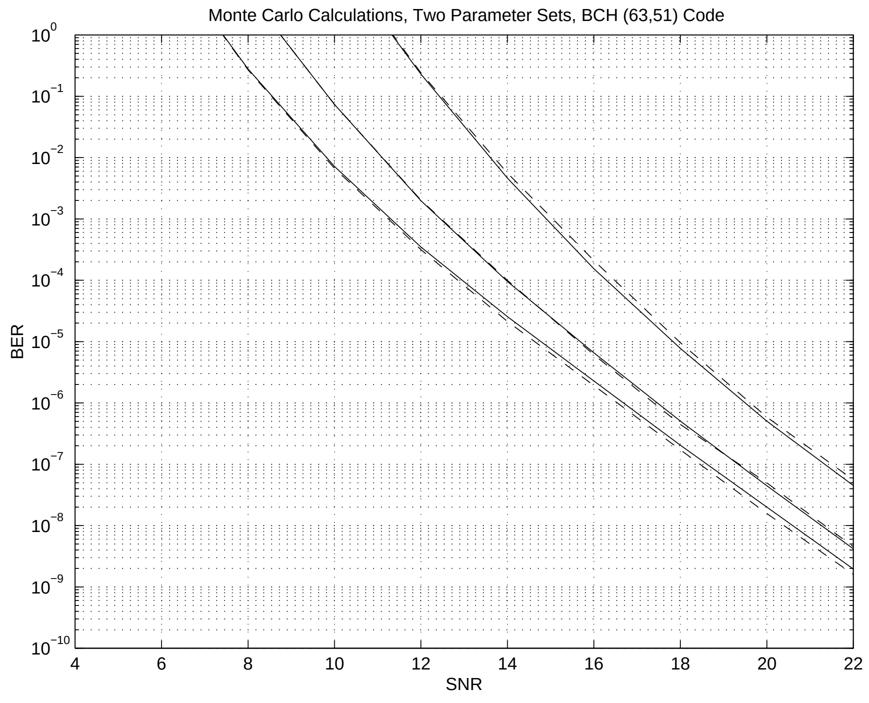

Figure 12: Monte Carlo (solid line) and reduced Monte Carlo (dashed line) for $\gamma = 0$ and $FT_c = 0.1$ (bottom pair), 0.03 (middle pair) and 0.01 (top pair).

The exponential correlation model used in the development has no basis in physical reality and is merely a mathematical convenience for the sake of tractability. A more realistic analysis would have to replace the exponential correlation model with Clarke's Bessel function correlation model, since Clarke's model is derived from the propagation of electromagnetic plane waves.

One possible effect of using Clarke's model is due to the fact that a Bessel function decays to zero more slowly than an exponential function. As a result, the **L** matrices will tend to be more singular and therefore have smaller determinants. Hence, for the same codeword, the denominator of (3.1), (3.4) and (3.5) will tend to be larger than for the exponential correlation model. This would increase that codeword's contribution to the total error probability. The net effect of this shift would have to be investigated.

The analysis also focussed on specific codes (in particular, Reed-Muller and BCH codes). A more rigorous analysis would have to analyze random codes. If the same general trends appeared when analyzing a random code, one would have a stronger reason to trust the methods.

The calculation of constants T_1 and T_2 also require a more rigorous foundation that would give some indication of the tradeoff between precision and the number of data points. One would expect to find that the precision is inversely proportional to the square root of the number of data points.

The method for determining when to stop adding codewords to the bins needs a firmer foundation. Terminating the search when the error sum changes by less than 2.5% to works well but hardly provides a rigorous bound. Given the nature of importance sampling, an exact bound or mathematical formulation may not be possible, but some sort of reliable mechanism to terminate the algorithm short of an exhaustive search and deliver an accurate result may exist.

CHAPTER IV

CALCULATIONS AND SIMULATIONS

The development of techniques to model and analyze the fading communication channel encountered in wireless communications systems has received considerable research interest and effort. Clarke's model, [33] which represents the received carrier signal as a sum of random-phase plane waves, serves as a popular starting point for investigations into such channels. This model, while quite useful for generating simulated fading waveforms,[1] does not facilitate the derivation of analytical closed form expressions for important items such as the distribution of fade durations. This limits its direct utility in determining error control code performance across fading channels.

The use of hidden Markov models (HMMs) to represent the channel alleviates these problems, as HMMs readily lend themselves to closed form calculations. In addition, the literature abounds with efficient techniques for fitting HMMs to experimental data. This chapter begins by presenting relevant material regarding Clarke's model, computer generation of fading waveforms, and traditional quantization and HMM modeling methods. A novel alternative quantization scheme and modeling techniques are then presented followed by comparisons of analytical HMM performance predictions with waveform level simulations.

These performance predictions are then used to determine the optimal error control code for a channel given QoS criteria such as BER, throughput, and delay. Finally, a novel approach for dynamically adjusting the error control code to track changes in channel conditions is discussed. It uses short- and long-term estimates of channel SNR to estimate whether the SNR is increasing or decreasing and picks an appropriate code for the channel.

[1]Clarke's model is often called Jakes' model when used in this manner

4.1 Fading

Many wireless channels, such as those encountered when analyzing macrocellular systems, are characterized by a narrowband signal and the absence of a Line Of Sight (LOS) path between the transmitter and receiver. The Rayleigh fading model matches empirical measurements of such systems very well. It assumes that the magnitude of the received carrier signal envelope has a Rayleigh distribution:

$$p_{|r|}(x) = \frac{x}{\sigma^2} e^{-\frac{x^2}{2\sigma^2}} \tag{4.1}$$

where σ^2 is the variance of $|r(t)|$. The received power (the square of the envelope, $|r(t)|^2$) therefore has an exponential distribution:

$$P_{|r|^2}(x) = \frac{1}{2\sigma^2} e^{-\frac{x}{2\sigma^2}} \tag{4.2}$$

and an average value of $2\sigma^2$, i.e., $E\{|r(t)|^2\} = 2\sigma^2$. These distributions will prove useful in determining quantization intervals for a Rayleigh fading waveform.

Jakes' method for simulating Rayleigh fading [34] serves as an excellent algorithm for generating such waveforms on a computer. However, when used to produce multiple waveforms, the waveforms exhibit considerable cross correlation.

One method of solving this problem uses the mutually orthogonal columns from a Walsh-Hadamard matrix[2] of order N_0; the entries in the columns are used to weight the oscillators that are summed to create the final waveforms, effectively passing the oscillator outputs through a Walsh Transform [35]. Thus, N_0 oscillators and an order N_0 ($N_0 \times N_0$) Walsh-Hadamard matrix $\mathbf{W}_{N_0}$ can produce up to N_0 uncorrelated Rayleigh fading waveforms. The autocorrelation of the generated waveforms tends to match the ideal, $J_0(2\pi f_D \tau)$, more closely as the number of oscillators increases.

Some restrictions exist on the choice of N_0; one cannot create a Walsh-Hadamard matrix for any N_0 picked out of thin air[3]. It would be convenient for $\mathbf{W}_{N_0}$ to exist for N_0 equal to

[2]A Walsh-Hadamard matrix $\mathbf{W}$ of order N_0 is an $N_0 \times N_0$ matrix $\mathbf{W}_{N_0}$ whose entries equal ± 1 arranged such that $\mathbf{W}_{N_0}\mathbf{W}_{N_0}^T = N_0\mathbf{I}$ [30].

[3]A Walsh-Hadamard matrix must have order $N_0 \in \{1, 2, 4 \cdot n\}, n \in \mathbb{Z}^+$. This is not a guarantee that such

powers of two since computers naturally operate on powers of two. Fortunately, not only do such they exist, but Walsh-Hadamard matrices with orders a power of two are quite easy to construct. The construction proceeds recursively: [30]

$$\mathbf{W}_{2^n} = \underbrace{\mathbf{W}_2 \otimes \mathbf{W}_2 \otimes \cdots \otimes \mathbf{W}_2}_{\text{n times}} \quad \mathbf{W}_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \tag{4.3}$$

For example,

$$\mathbf{W}_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{W}_8 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 & 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 & 1 & -1 & -1 & 1 \\ 1 & 1 & 1 & 1 & -1 & -1 & -1 & -1 \\ 1 & -1 & 1 & -1 & -1 & 1 & -1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & 1 & 1 \\ 1 & -1 & -1 & 1 & -1 & 1 & 1 & -1 \end{bmatrix}$$

The procedure suggested in [35] for generating waveform m of N_0 then proceeds as follows:

$$\begin{aligned} R(m,t) &= \sqrt{\tfrac{2}{N_0}} \sum_{n=1}^{N_0} w_{m,n} \big(cos(\beta_n) + j sin(\beta_n)\big) cos(\omega_n t + \theta_n) \\ w_{m,n} &= \text{Entry from } N_0 \text{ by } N_0 \text{ Walsh-Hadamard matrix } \mathbf{W} \\ \beta_n &= \pi n / N_0 \\ \theta_n &= \text{Random value} \in [0 \cdots 2\pi) \end{aligned} \tag{4.4}$$

Figure 13 shows a typical realization of (4.4). The normalization factor of $\sqrt{\frac{2}{N_0}}$ gives the waveform an average power of one, which corresponds to $\sigma^2 = 0.5$. Figure 14 displays the PDF of the Rayleigh fading waveform and the PDF of the instantaneous received power.

To produce N_1 waveforms, where $N_1 < N_0$, one should sum groups of N_0/N_1 oscillators and then multiply the results by an entry from $\mathbf{W}_{N_1}$ as in (4.4). The groups should be

a matrix exists!

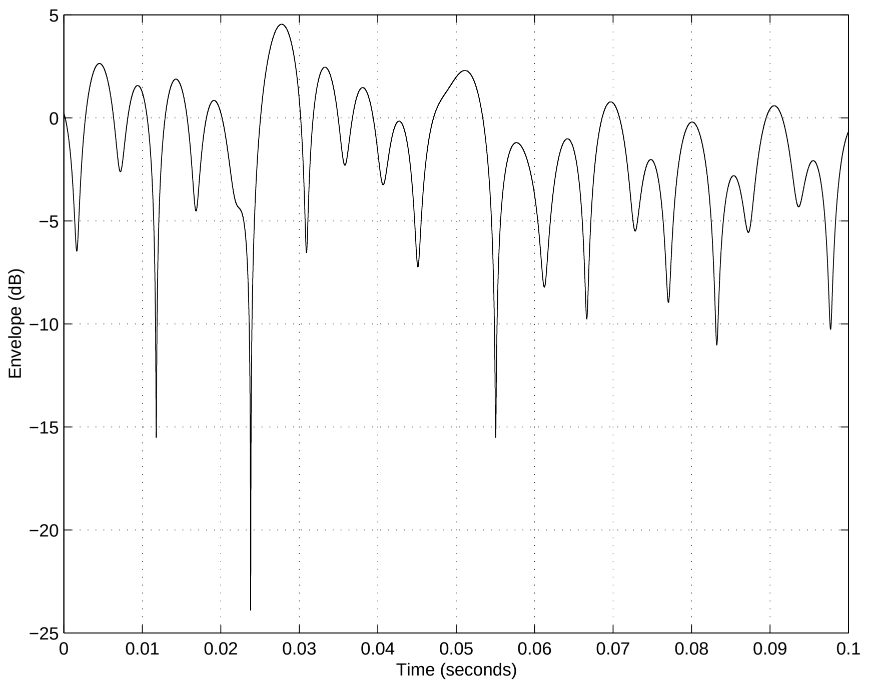

Figure 13: Rayleigh Fading Envelope for 1.2 GHz, 100 km/h, $N_0 = 64$

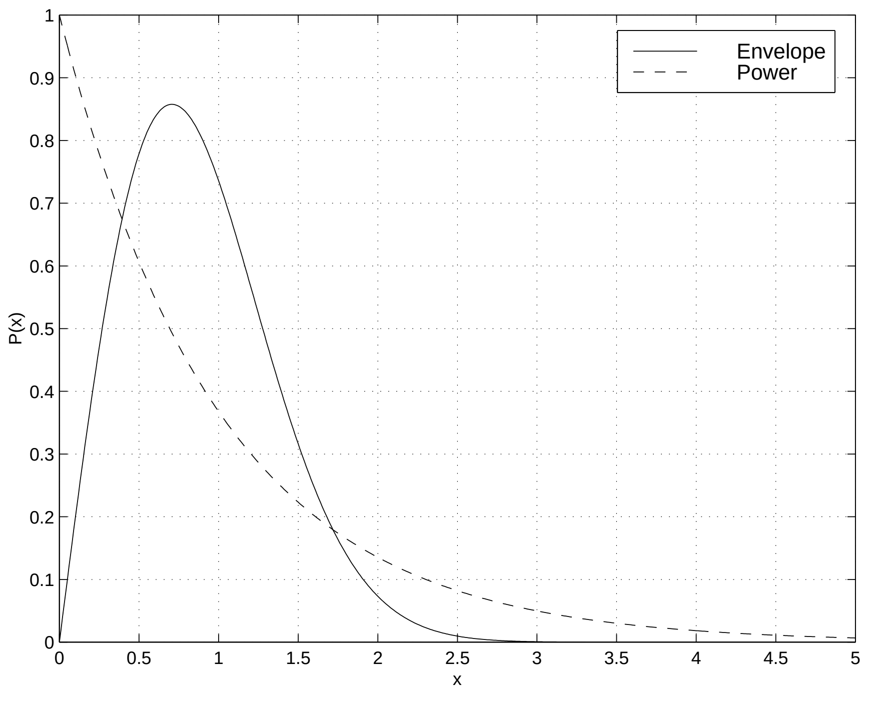

Figure 14: Rayleigh Envelope and Power CDFs for $\sigma^2 = 0.5$

chosen evenly from the received Doppler spectrum, e.g., if $N_0 = 64$ and $N_1 = 16$, choose 1+17+33+49, 2+18+34+50, etcetera.

The existence of a Line of Sight (LOS) component changes the statistics of the received envelope considerably. The received envelope then has a Rice distribution instead of a Rayleigh distribution. A Rice distribution has an additional parameter, referred to as the Rice factor and typically denoted K, that specifies the ratio in power between the LOS (l^2) and diffuse ($2\sigma^2$) components of the received waveform:

$$K = \frac{l^2}{2\sigma^2} \tag{4.5}$$

One can simulate Rician fading by adding appropriately scaled LOS and Rayleigh fading components together.

The distribution of the received envelope and power can be expressed in terms of the of the power in the LOS and diffuse components or in terms of the Rice factor K:

$$\begin{aligned} p_{|r|}(x) &= \tfrac{x}{\sigma^2} e^{-\left(\frac{x^2+l^2}{2\sigma^2}\right)} I_0\left(\tfrac{xl}{\sigma^2}\right) \\ &= \tfrac{2x(K+1)}{s^2+2\sigma^2} e^{-\left(K+\frac{(K+1)x^2}{s^2+2\sigma^2}\right)} I_0\left(2x\sqrt{\tfrac{K(K+1)}{s^2+2\sigma^2}}\right) \end{aligned} \tag{4.6}$$

$$\begin{aligned} p_{|r|^2}(x) &= \tfrac{1}{2\sigma^2} e^{-\left(\frac{l^2+x}{2\sigma^2}\right)} I_0\left(\tfrac{l\sqrt{x}}{\sigma^w}\right) \\ &= \tfrac{K+1}{s^2+2\sigma^2} e^{-\left(K+\frac{(K+1)x}{s^2+2\sigma^2}\right)} I_0\left(2\sqrt{\tfrac{K(K+1)x}{s^2+2\sigma^2}}\right) \end{aligned} \tag{4.7}$$

The Rice Envelope PDF for several values of K appears in Figure 15. The mathematically inclined will notice that the equations for the Rice PDFs reduce to those for the Rayleigh PDFs when $K = 0$.

The results of Rayleigh and Rice fading models do not agree well with empirical measurements when the range in propagation delays approaches or exceeds exceeds the inverse channel bandwidth. In such cases, it is necessary to model individual delayed arrivals to produce sufficiently accurate simulation results.

The structure for modeling these arrivals resembles the tapped delay line used to perform Finite Impulse Response (FIR) filtering (see Figure 16). It differs in that the filter coefficients vary in time, typically exhibit Rayleigh or Rice fading profiles, and are scaled

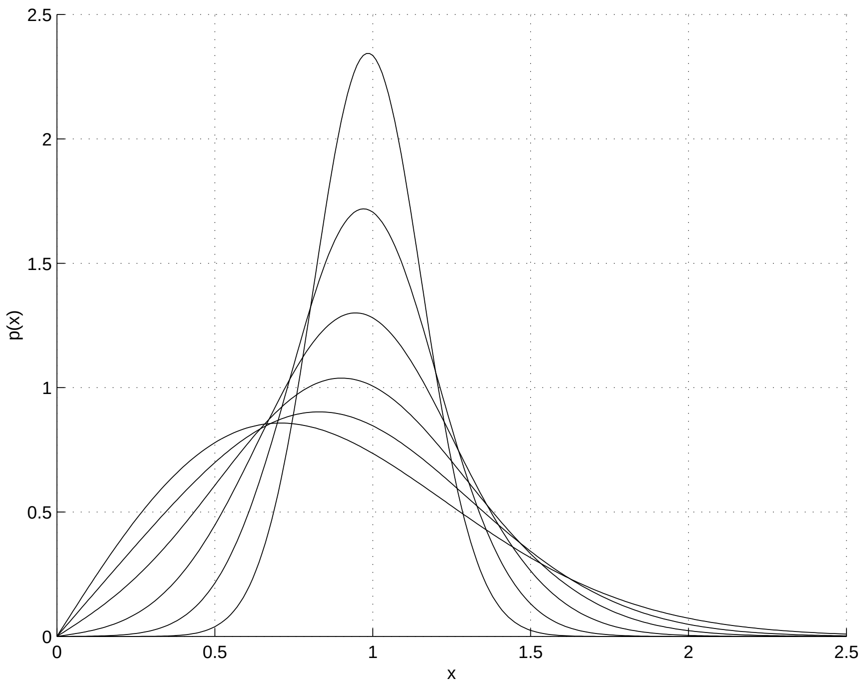

Figure 15: Rice Envelope PDF for K=0 (lowest peak), 1, 2, 4, 8, 16 (highest peak)

so that the total average received power equals one. The value of τ, which represents the simulation time step interval, varies with the application.

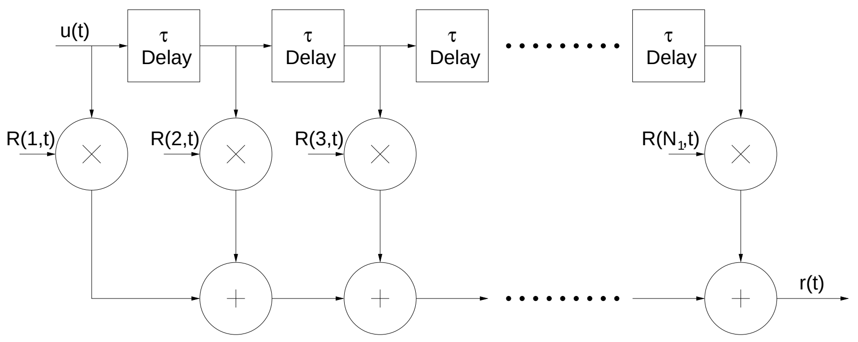

Figure 16: The τ-spaced channel

The COST 207 channel studies [28], which were undertaken during the development of the GSM cellular system, resulted in several sets of recommended coefficients for simulating macrocellular radio propagation. The different sets of coefficients covered multiple terrain types, such as rural, hilly terrain and urban environments. Another set of coefficients provided an equalization test pattern. The study gave two sets of six and two sets of twelve tap coefficients for each environment. As an example, a set of six and twelve tap coefficients

Table 5: Two COST 207 Typical Urban Delay Profiles

Delay, μs	Avg. Rel. Power (db)	Avg. Rel. Power
0.0	-4.0	0.0921
0.1	-3.0	0.1159
0.3	0.0	0.2313
0.5	-2.6	0.1271
0.8	-3.0	0.1159
1.1	-5.0	0.0732
1.3	-7.0	0.0462
1.7	-5.0	0.0731
2.3	-6.5	0.0518
3.1	-8.6	0.0319
3.2	-11.0	0.0184
5.0	-10.0	0.0231

Delay, μs	Avg. Rel. Power (db)	Avg. Rel. Power
0.0	-3.0	0.1897
0.2	0.0	0.3785
0.5	-2.0	0.2388
1.6	-6.0	0.0951
2.3	-8.0	0.0600
5.0	-10.0	0.0379

for the typical urban are listed in Table 5 and shown graphically in Figure 17.

These values scale the fading waveform that multiplies the taps in the τ-spaced channel. The values in Table 5 appear to indicate a value for τ of $0.1\mu s$. One may, however, choose a larger value for τ and adjust the delays accordingly. For example, given the twelve ray typical urban delay profile and a desired τ of $0.5\mu s$, one could combine the $0.0\mu s$ and $0.1\mu s$ rays, the $0.3\mu s$ and $0.5\mu s$ rays, and so on.

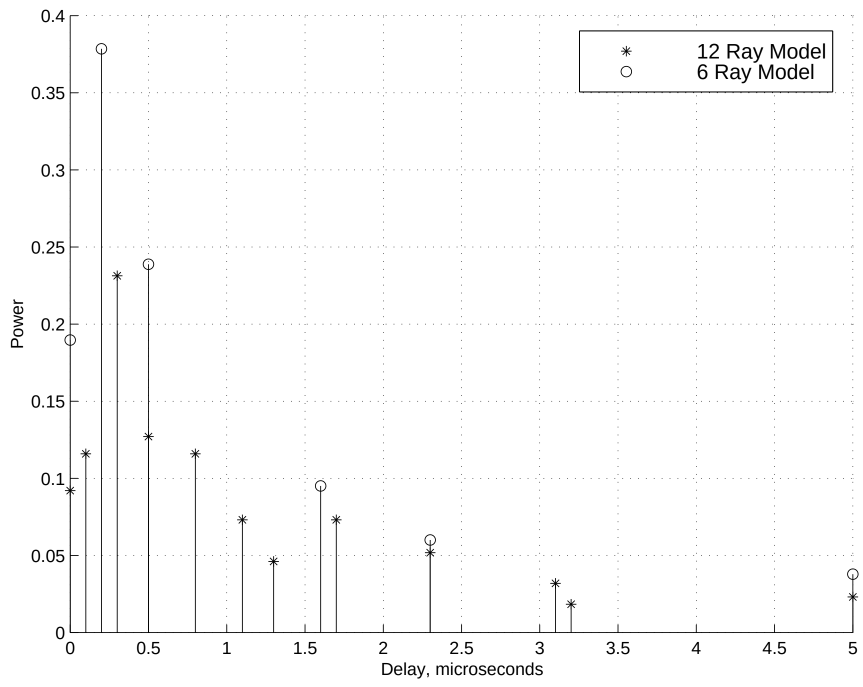

Figure 17: Two COST 207 Typical Urban Delay Profiles

4.2 Quantizing

A quantizer converts a series of infinite precision discrete time values into another sequence of discrete time values where each sample has finite precision. Designing a quantizer involves several design decisions, including choosing the number of quantizing intervals, the boundaries of the quantization intervals, and the value to map entries in the quantization interval to.

Let L represent the number of quantization levels; $x_0, x_1, \ldots, x_K$ the boundaries of the quantization intervals; and $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_K$ the quantized values for inputs in the ranges $[x_0, x_1), [x_1, x_2), \ldots, [x_{K-1}, x_K]$; respectively.

Given an input signal PDF $p(x)$ and a quantization error function $f(x)$, the quantization distortion D for an L level quantizer is then:

$$D = \sum_{k=1}^{L} \int_{x_{k-1}}^{x_k} f(\tilde{x}_k - x)p(x) \tag{4.8}$$

Since a Rayleigh PDF is beging quantized, choose $x_0 = 0$ and $x_L = +\inf$ and $p(x) = \frac{x}{\sigma^2}\exp\{-x^2/2\sigma^2\}$. The natural approach in choosing the x_ks and $\tilde{x}_k$s involves differentiating (4.8) with respect to x_k and $\tilde{x}_k$. This yields $2L-1$ equations:

$$f(\tilde{x}_k - x_k) = f(\tilde{x}_{k+1} - x_k), \quad k = 1, 2, \ldots, L-1 \tag{4.9}$$

$$\int_{x_{k-1}}^{x_k} f'(\tilde{x}_k - x)p(x)dx = 0, \quad k = 1, 2, \ldots, L \tag{4.10}$$

This analysis will use the squared error criterion, so $f(x) = x^2$. and (4.9) and (4.10) may be rewritten as:

$$x_k = \frac{1}{2}(\tilde{x}_k + \tilde{x}_{k+1}), \quad k = 1, 2, \ldots, L \tag{4.11}$$

$$\int_{x_{k-1}}^{x_k} (\tilde{x}_k - x)p(x)dx = 0, \quad k = 1, 2, \ldots, L \tag{4.12}$$

Consider the case of Rayleigh fading (where $p(x)$ has a Rayleigh distribution) which requires solving (4.11) and (4.12) to find the quantization regions and the value to which all of the samples in the quantization region are mapped. A graphical display of the results for 8- and 64-level quantization appears in Figures 18 and 19, respectively, and the numerical results in 6. The results for the 8-level case can be seen to agree with those in [24].

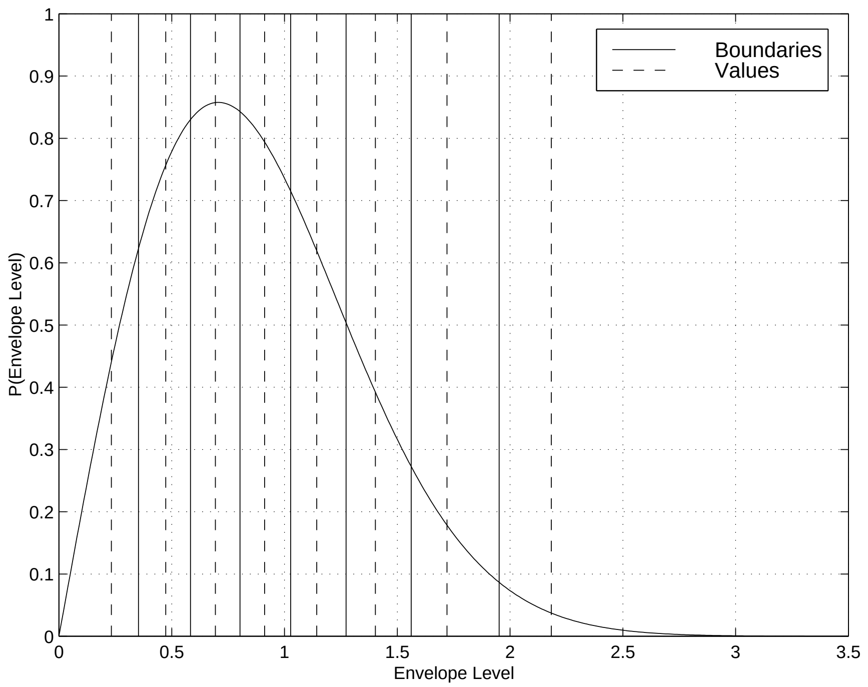

Figure 18: MMSE 8-Level Quantization for a Rayleigh PDF

This dissertation will soon demonstrate that MMSE quantization suffers from severe shortcomings when used to generate hidden Markov models and that a logarithmic quantization scheme produces far more accurate results. Several logarithmic quantization schemes are then presented as a first attempt at capturing the deep fades present in Rayleigh fading, but no claims are made with regard to their optimality.

Once it was determined that MMSE quantization did not capture the fades of the Rayleigh facing process well, the shape of the Rayleigh PDF was examined in an attempt to devise a more appropriate scheme. The idea for a logarithmic quantization scheme came about while examining the shape of the Rayleigh PDF. The tail of the Rayleigh PDF has a linear decay when plotted on a logarithmic scale as in Figure 20.

A search of the literature found one paper that made use of logarithmic quantization intervals; the authors in [1] chose the intervals (0, 0.0149, 0.0329, 0.0725, 0.1601, 0.3535,

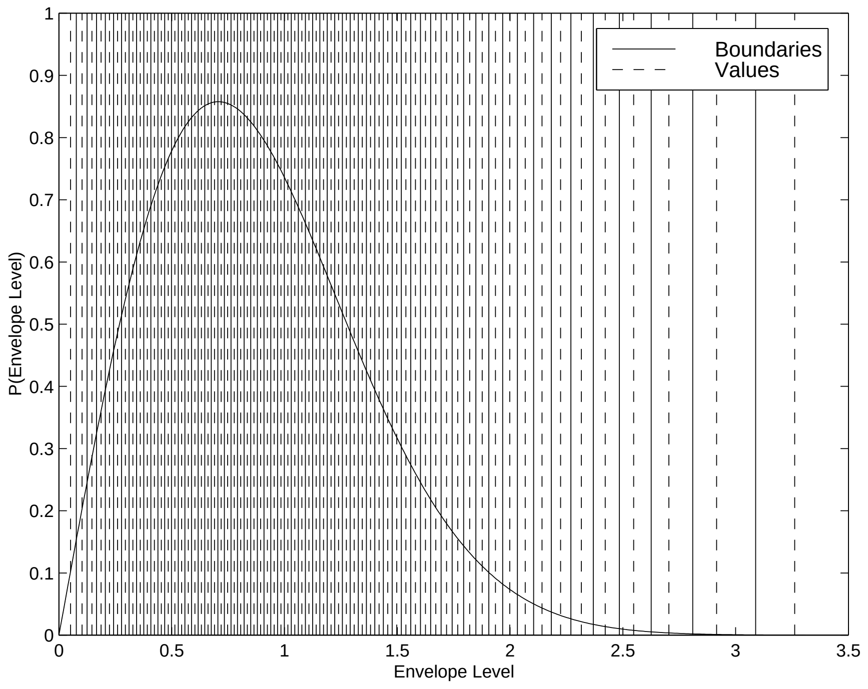

Figure 19: MMSE 64-Level Quantization for a Rayleigh PDF

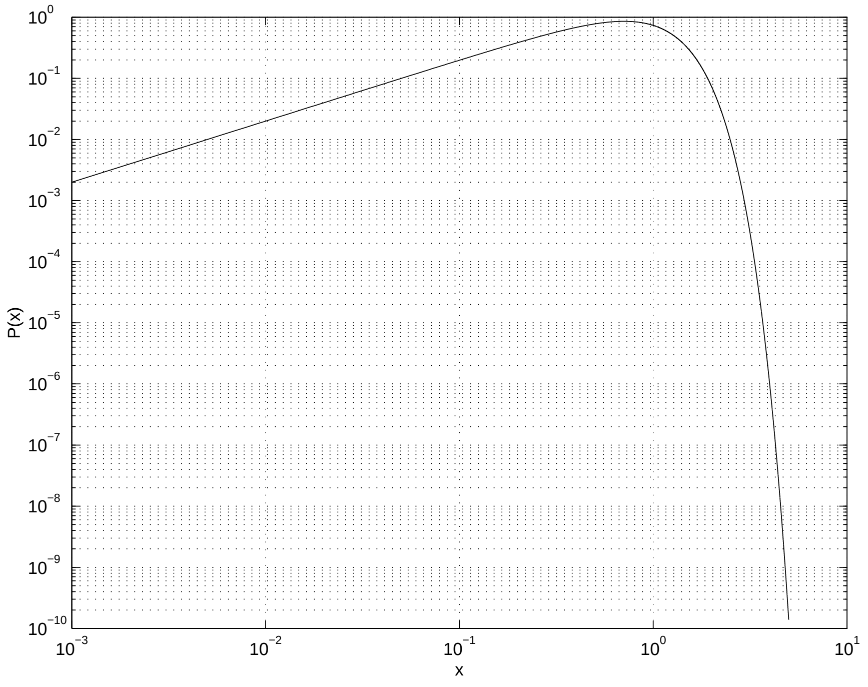

Figure 20: Logarithmic plot of a Rayleigh PDF

Table 6: Quantization Intervals for 8 and 64-Level MMSE Quantization

8-Level Quantization

k	x_k	$\tilde{x}_k$
1	0.3530	0.2324
2	0.5834	0.4735
3	0.8027	0.6933
4	1.0276	0.9120
5	1.2728	1.1431
6	1.5615	1.4026
7	1.9514	1.7203
8	$+\infty$	2.1824

64-Level Quantization

k	x_k	$\tilde{x}_k$	k	x_k	$\tilde{x}_k$
1	0.0770	0.0513	33	1.0926	1.0768
2	0.1246	0.1026	34	1.1244	1.1084
3	0.1667	0.1466	35	1.1567	1.1405
4	0.2055	0.1867	36	1.1895	1.1730
5	0.2422	0.2243	37	1.2229	1.2061
6	0.2773	0.2601	38	1.2569	1.2397
7	0.3112	0.2945	39	1.2915	1.2740
8	0.3441	0.3279	40	1.3269	1.3090
9	0.3762	0.3603	41	1.3631	1.3448
10	0.4077	0.3921	42	1.4002	1.3814
11	0.4386	0.4233	43	1.4383	1.4190
12	0.4691	0.4540	44	1.4774	1.4576
13	0.4993	0.4843	45	1.5178	1.4973
14	0.5291	0.5142	46	1.5595	1.5383
15	0.5587	0.5439	47	1.6026	1.5807
16	0.5881	0.5734	48	1.6475	1.6246
17	0.6173	0.6027	49	1.6942	1.6703
18	0.6464	0.6319	50	1.7430	1.7180
19	0.6755	0.6609	51	1.7943	1.7680
20	0.7045	0.6900	52	1.8484	1.8206
21	0.7335	0.7190	53	1.9057	1.8762
22	0.7625	0.7480	54	1.9669	1.9353
23	0.7916	0.7770	55	2.0327	1.9985
24	0.8208	0.8062	56	2.1041	2.0668
25	0.8501	0.8354	57	2.1824	2.1413
26	0.8796	0.8648	58	2.2697	2.2236
27	0.9092	0.8943	59	2.3689	2.3159
28	0.9390	0.9241	60	2.4847	2.4219
29	0.9691	0.9540	61	2.6256	2.5475
30	0.9995	0.9843	62	2.8096	2.7037
31	1.0302	1.0148	63	3.0884	2.9155
32	1.0612	1.0456	64	$+\infty$	3.2613

0.7805, 1.7234, ∞) which are spaced approximately 3.4dB apart.

Several several logarithmic quantization schemes were investigated:

The first quantization used the point at which the Rayleigh distribution attains its maximum value as an endpoint for one of the quantization intervals. This value equals:

$$\begin{aligned} \tfrac{d}{dx}2xe^{-x^2} &= 0 \\ 2e^{-x^2} - 4x^2e^{-x^2} &= 0 \end{aligned}$$

$$\text{which is satisfied for } x = \pm\sqrt{2}/2$$

Each of the eight quantization intervals then covered 3.0 or 3.5dB. $\sqrt{2}/2$ corresponds almost exactly to -1.5dB, which gives quantization intervals equal to:

(-16.5, -13.5, -10.5, -7.5, -4.5, -1.5, 1.5, ∞)

(-19, -15.5, -12, -8.5, -5, -1.5, 2, ∞)

Another quantization scheme used the average value of the Rayleigh distribution as a quantization interval endpoint. The average value equals:

$$\int_{0}^{+\infty} x \cdot 2xe^{-x^2}dx = \sqrt{\pi}/2$$

which calculates out to 0.8862 or approximately -0.5246dB. Given quantization intervals that span 3 or 3.5dB, the intervals equal:

(-15.5246, -12.5246, -9.5246, -6.5246, -3.5246, -0.5246, 2.4754, ∞)

(-18.0246, -14.5246, -11.0246, -7.5246, -4.0246, -0.5246, 2.9754, ∞)

A quantization scheme similar to the one described in [1] was also examined. This scheme uses eight uniform quantization intervals that cover 3.5dB-wide regions. The interval endpoints include -1 dB as a reference point, which gives intervals of:

(-18.5, -15, -11.5, -8, -4.5, -1, 2.5, ∞) or (0, 0.0141, 0.0316, 0.0708, 0.1585, 0.3548, 0.7943, 1.7783, ∞)

These intervals, as well as their logarithmic centers, appear in Table 4.2 and Figure 21. These intervals also appear against a logarithmic plot of the Rayleigh PDF in Figure 22.

16, 32 and 64-level quantization schemes were calculated by breaking each of the intervals in half 2, 4 or 8 times. This methodology added additional intervals above the highest and

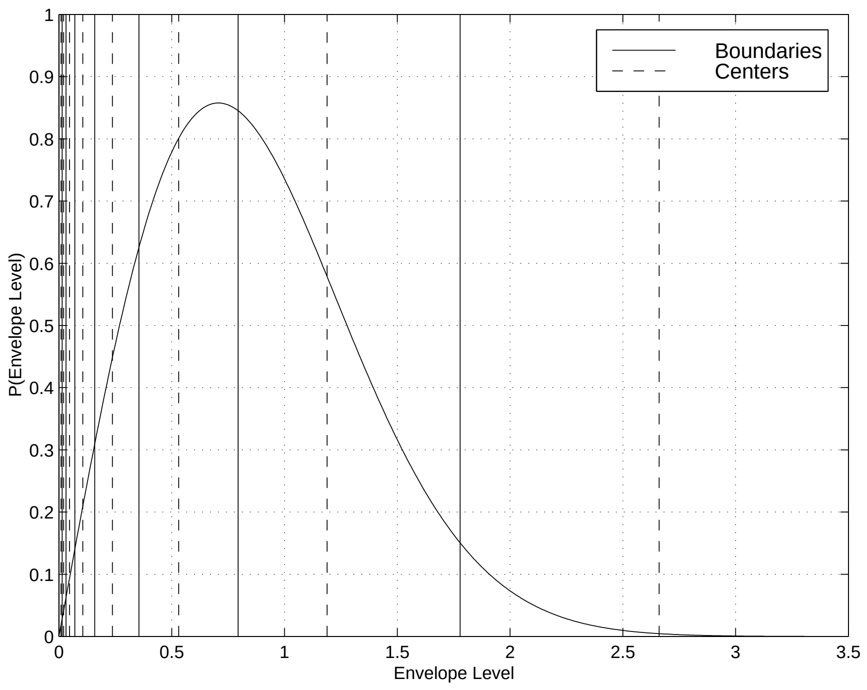

Figure 21: Logarithmic 8-Level Quantization for a Rayleigh PDF

below the lowest interval during each iteration. The results for the 64-level scheme appear in Table 8 and Figure 23.

Table 7: Quantization Intervals for 8-Level Logarithmic Quantization

k	x_k	dB	$\tilde{x}_k$	dB
1	0.0141	-18.5000	0.0094	-20.2500
2	0.0316	-15.0000	0.0211	-16.7500
3	0.0708	-11.5000	0.0473	-13.2500
4	0.1585	-8.0000	0.1059	-9.7500
5	0.3548	-4.5000	0.2371	-6.2500
6	0.7943	-1.0000	0.5309	-2.7500
7	1.7783	2.5000	1.1885	0.7500
8	$+\infty$	$+\infty$	2.6607	4.2500

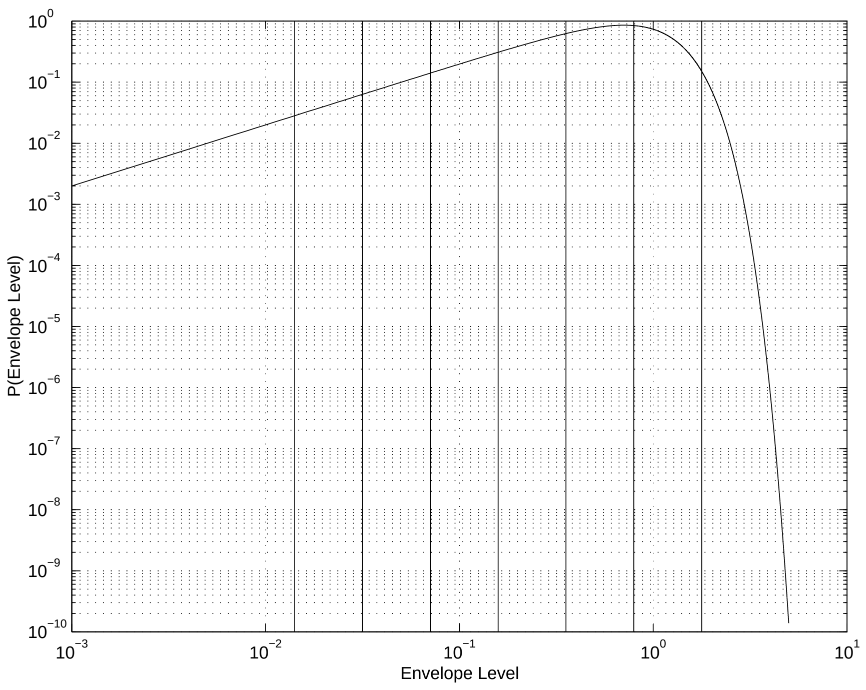

Figure 22: Logarithmic 8-Level Quantization for a Rayleigh PDF

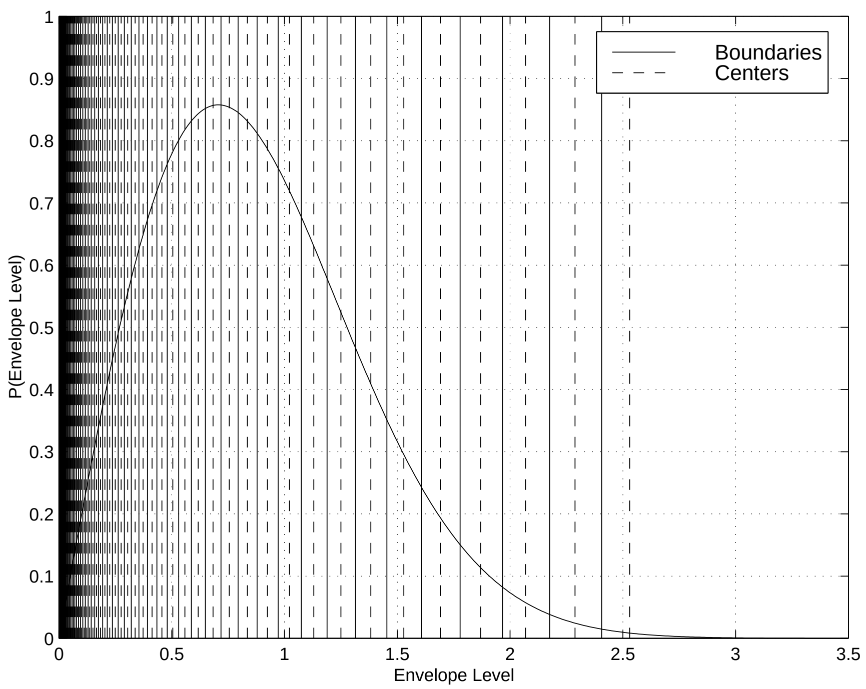

Figure 23: Logarithmic 64-Level Quantization for a Rayleigh PDF

Table 8: Quantization Intervals for 64-Level Logarithmic Quantization

k	x_k	dB	$\tilde{x}_k$	dB
1	0.0047	-23.312	0.0044	-23.5312
2	0.0052	-22.875	0.0049	-23.0938
3	0.0057	-22.437	0.0054	-22.6562
4	0.0063	-22.000	0.0060	-22.2188
5	0.0070	-21.562	0.0066	-21.7812
6	0.0077	-21.125	0.0073	-21.3438
7	0.0085	-20.687	0.0081	-20.9062
8	0.0094	-20.250	0.0090	-20.4688
9	0.0104	-19.812	0.0099	-20.0312
10	0.0115	-19.375	0.0110	-19.5938
11	0.0128	-18.937	0.0121	-19.1562
12	0.0141	-18.500	0.0134	-18.7188
13	0.0156	-18.062	0.0149	-18.2812
14	0.0173	-17.625	0.0164	-17.8438
15	0.0191	-17.187	0.0182	-17.4062
16	0.0211	-16.750	0.0201	-16.9688
17	0.0234	-16.312	0.0222	-16.5312
18	0.0259	-15.875	0.0246	-16.0938
19	0.0286	-15.437	0.0272	-15.6562
20	0.0316	-15.000	0.0301	-15.2188
21	0.0350	-14.562	0.0333	-14.7812
22	0.0387	-14.125	0.0368	-14.3438
23	0.0428	-13.687	0.0407	-13.9062
24	0.0473	-13.250	0.0450	-13.4688
25	0.0523	-12.812	0.0498	-13.0312
26	0.0579	-12.375	0.0550	-12.5938
27	0.0640	-11.937	0.0609	-12.1562
28	0.0708	-11.500	0.0673	-11.7188
29	0.0783	-11.062	0.0745	-11.2812
30	0.0866	-10.625	0.0823	-10.8438
31	0.0958	-10.187	0.0911	-10.4062
32	0.1059	-9.7500	0.1007	-9.9688

k	x_k	dB	$\tilde{x}_k$ dB	
33	0.1172	-9.3125	0.1114	-9.5313
34	0.1296	-8.8750	0.1232	-9.0938
35	0.1433	-8.4375	0.1363	-8.6563
36	0.1585	-8.0000	0.1507	-8.2188
37	0.1753	-7.5625	0.1667	-7.7813
38	0.1939	-7.1250	0.1843	-7.3438
39	0.2144	-6.6875	0.2039	-6.9063
40	0.2371	-6.2500	0.2255	-6.4688
41	0.2623	-5.8125	0.2494	-6.0313
42	0.2901	-5.3750	0.2758	-5.5938
43	0.3208	-4.9375	0.3051	-5.1563
44	0.3548	-4.5000	0.3374	-4.7188
45	0.3924	-4.0625	0.3731	-4.2813
46	0.4340	-3.6250	0.4127	-3.8438
47	0.4800	-3.1875	0.4564	-3.4063
48	0.5309	-2.7500	0.5048	-2.9688
49	0.5872	-2.3125	0.5583	-2.5313
50	0.6494	-1.8750	0.6175	-2.0938
51	0.7182	-1.4375	0.6829	-1.6563
52	0.7943	-1.0000	0.7553	-1.2188
53	0.8785	-0.5625	0.8354	-0.7813
54	0.9716	-0.1250	0.9239	-0.3438
55	1.0746	0.3125	1.0218	0.0938
56	1.1885	0.7500	1.1301	0.5313
57	1.3145	1.1875	1.2499	0.9688
58	1.4538	1.6250	1.3824	1.4063
59	1.6079	2.0625	1.5289	1.8438
60	1.7783	2.5000	1.6909	2.2813
61	1.9668	2.9375	1.8701	2.7188
62	2.1752	3.3750	2.0684	3.1563
63	2.4057	3.8125	2.2876	3.5938
64	$+\infty$	$+\infty$	2.5301	4.0313

4.3 Error Rate

This section investigate the assignment of error probabilities to quantization intervals for the case of binary phase shift keying (BPSK) modulation in the presence of additive white Gaussian noise (AWGN). Each bit has power E_b, which will be normalized to one, and Gaussian noise having power N_0 (which implies that the noise samples will describe a Gaussian PDF having $\sigma^2 = N_0/2$).

The signal to noise ratio (SNR) specifies the ratio of the received signal power to the received noise power. Given a signal to noise ratio of E_b/N_0, the probability of receiving a transmitted bit in error for BPSK is:

$$P_e = Q\left(\sqrt{2\frac{E_b}{N_0}}\right) = \frac{1}{2}\,\mathrm{erfc}\left(\sqrt{\frac{E_b}{N_0}}\right) \tag{4.13}$$

A graph of this function appears in Figure 24. One should note that the BER decreases at a faster rate as the SNR increases, i.e., the magnitude of the first derivitive of the SNR vs. BER curve increases as the SNR increases. Hence, as the SNR increases, each additional increase results in a greater and greater decrease in the BER.

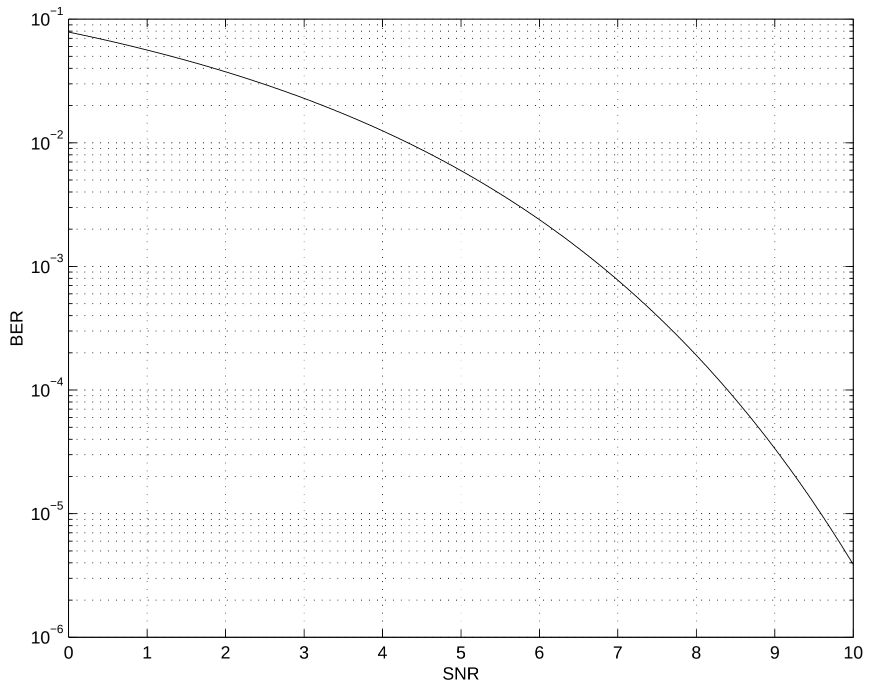

Figure 24: Bit Error Rate for Binary Phase Shift Keying

To calculate the BER in a Rayleigh fading environment, one must compute:

$$BER = \int_0^{+\infty} P(\gamma)P_e(\gamma)d\gamma \tag{4.14}$$

where γ is the magnitude of the received signal envelope, $P_e(\gamma)$ is the probability of decoding a bit erroneously given an envelope magnitude of γ, and $P(\gamma)$ is the probability of the envelope having magnitude γ.

The calculations progress most easily if one chooses the parameters of the Rayleigh fading process to give an average envelope power of one and adjust the received noise power to give the desired SNR. Progressing with this in mind, $P(\gamma)$ comes directly from (4.1), the expression for the Rayleigh PDF. A Rayleigh PDF must have a variance of 1/2 to give an average power of one, which gives:

$$P(\gamma) = 2\gamma e^{-\gamma^2} \tag{4.15}$$

The average SNR ($\overline{SNR}$) follows as:

$$\overline{SNR} = \frac{\overline{E_b}}{N_0} = \frac{1}{2\sigma_n^2} \tag{4.16}$$

where σ_n^2 is the variance of the AWGN noise process. Labeling the level of the received envelope γ gives an envelope power of γ^2. The instantaneous SNR can then be calculated as:

$$SNR = \gamma^2\, \overline{SNR} = \gamma^2 \frac{\overline{E_b}}{N_0} = \frac{\gamma^2}{2\sigma_n^2} \tag{4.17}$$

Substituting (4.17) into the BER equation (4.13) to get:

$$P_e(\gamma) = \frac{1}{2}\,\mathrm{erfc}\left(\sqrt{\gamma^2 \frac{\overline{E_b}}{N_0}}\right) = \frac{1}{2}\,\mathrm{erfc}\left(\gamma\sqrt{\frac{\overline{E_b}}{N_0}}\right) \tag{4.18}$$

and combine (4.15) and (4.18) as in (4.14) to get:

$$BER = \int_0^{+\infty} \gamma e^{-\gamma^2} \cdot \mathrm{erfc}\left(\sqrt{\gamma^2 \frac{\overline{E_b}}{N_0}}\right) d\gamma \tag{4.19}$$

Performing the integration gives:

$$BER = \frac{1}{2}\left(1 - \sqrt{\frac{\overline{E_b}/N_0}{\overline{E_b}/N_0 + 1}}\right) \tag{4.20}$$

A graph of (4.20) appears in Figure 25. Note that, unlike in the case of an AWGN channel in Figure 24), the BER falls off at a constant rate as the SNR increases. In fact, based on the slope of the line, one can see that doubling the transmitter power serves only to halve the BER.

The result in (4.20) stands as one of the few analytical results that one can calculate regarding the statistics of the Rayleigh fading channel. It will be used as a benchmark for gauging the accuracy of HMMs calculated in later sections.

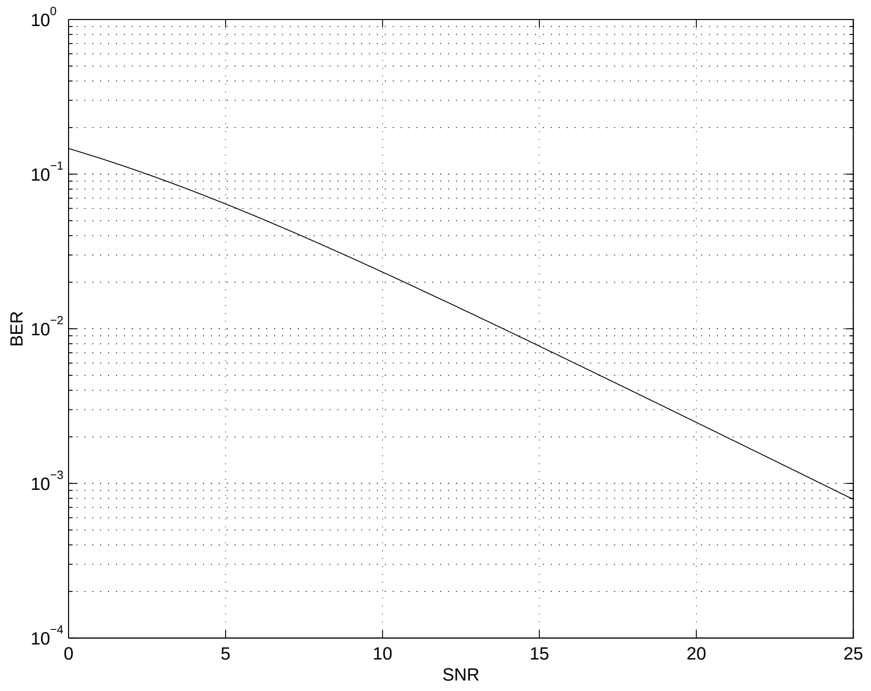

Figure 25: Bit Error Rate for Binary Phase Shift Keying in Rayleigh Fading

Once quantization intervals have been chosen, the next step is to calculate the error probability that will be associate with each each interval. This is accomplished by computing the average probability of error experienced in each interval. A first pass at this expression gives:

$$P_k = \frac{\frac{1}{2}\int_{x_{k-1}}^{x_k} \operatorname{erfc}\left(\gamma\sqrt{\frac{E_b}{N_0}}\right) d\gamma}{\int_{x_{k-1}}^{x_k} d\gamma} \tag{4.21}$$

Which simplifies to:

$$P_k = \frac{\frac{1}{2}\int_{x_{k-1}}^{x_k} \operatorname{erfc}\left(\gamma\sqrt{\frac{E_b}{N_0}}\right) d\gamma}{x_k - x_{k-1}} \tag{4.22}$$

However, the error probability for a quantization interval is more accurately calculated by weighting the the calculations by the Rayleigh PDF (an example appears in section 4.4):

$$P_k = \frac{\frac{1}{2}\int_{x_{k-1}}^{x_k} \gamma e^{-\gamma^2} \operatorname{erfc}\left(\gamma\sqrt{\frac{E_b}{N_0}}\right) d\gamma}{\int_{x_{k-1}}^{x_k} \gamma e^{-\gamma^2} d\gamma} \tag{4.23}$$

Note that the fade rate does not play a role in any these calculations. As an example, consider the fourth quantization interval from the 8-level quantization where the average SNR equals 10 dB. This interval covers the region from -8 dB (0.7079) to -11.5 dB (0.1585).

$$\begin{aligned} P_5 &= \frac{\int_{0.07079}^{0.1585} \gamma e^{-\gamma^2} \operatorname{erfc}\left(\gamma\sqrt{10}\right) d\gamma}{\int_{0.07079}^{0.1585} \gamma e^{-\gamma^2} d\gamma} \\ &= 0.2967 \end{aligned} \tag{4.24}$$

All of the quantization interval error probabilities, whether originating in a MMSE or a logarithmic scheme, are calculated in the same manner. Table 9 lists the error probabilities for the quantization regions in the 8-level MMSE and logarithmic schemes using (4.23). Note that all of the intervals in the MMSE quantization scheme have a fairly low probability of error, while the logarithmic scheme contains a wide range of error probabilities.

4.4 Markov Models

Several authors have investigated the modeling of a fading channel using a (non-hidden) Markov model [24] [22] [21]. The authors in [36] argued that this model adequately characterized the Rayleigh fading waveform. Similar models were tried initially but quickly found rather inadequate for detailed applications. These inadequacies are explained in this section.

Table 9: Error Probabilities for 8-Level MMSE and Logarithmic Quantization

8-Level MMSE Quantization

k	x_k	P_k
1	0.3530	.1654
2	0.5834	.0211
3	0.8027	.0014
4	1.0276	4.159e-05
5	1.2728	4.383e-07
6	1.5615	9.948e-10
7	1.9514	1.731e-13
8	$+\infty$	1.177-19

8-Level Logarithmic Quantization

k	x_k	P_k
1	0.0141	0.4832
2	0.0316	0.4573
3	0.0708	0.4052
4	0.1585	0.2967
5	0.3548	0.1228
6	0.7943	0.0109
7	1.7783	1.784e-05
8	$+\infty$	8.18e-17

The 3-tuple $(S, \boldsymbol{\pi}, \mathbf{P})$ describes a Markov model, where N to denotes the number of states in the model, and the remaining symbols represent represent the following sets:

- S is the set of Markov states: $S = (s_1, s_2, \ldots, s_N)$,
- $\boldsymbol{\pi}$ is the vector of initial state probabilities: $\pi_i = P(s(0) = i)$
- $\mathbf{P}$ is the matrix of state transition probabilities: $p_{i,j} = P(s(n+1) = j | s(n) = i)$.

A few quick calculations demonstrate the problems encountered when attempting to use this model for serious work. The following example involves a normalized fade rate of 0.01 and an 8-level MMSE quantization scheme. The Markov transition probabilities were calculated using Monte Carlo methods: fifty instances of the the Rayleigh fading process $r(t)$ were simulated (each instance seeded the local oscillators with different initial values) and 5,000,000 samples of each instance's amplitude $|r(nT)|$ were used to calculate $n_{i,j}$, the number of transitions from state (quantization interval) i to state (quantization interval) j. This gives Markov transition matrix entries:

$$p_{i,j} = n_{i,j}/n_i, \quad n_i = \sum_{j=1}^{N} n_{i,j} \tag{4.25}$$

These fifty matrices were then averaged together to mitigate the effects of parameter choice while keeping track of the standard deviation of each of the resulting matrix entries. The

Table 10: 8-State Markov Model results, unscaled, $FT_c = 0.01$

SNR	Analytical BER	Markov BER MMSE	Markov BER logarithmic
0	0.1464	0.1508	0.1410
5	0.0642	0.0709	0.0654
10	0.0233	0.0321	0.0247
15	0.0077	0.0168	0.0084
20	0.0025	0.0094	0.0027

Table 11: 8-State Markov Model results, scaled, $FT_c = 0.01$

SNR	Analytical BER	Markov BER MMSE	Markov BER logarithmic
0	0.1464	0.1473	0.1465
5	0.0642	0.0645	0.0640
10	0.0233	0.0234	0.0231
15	0.0077	0.0078	0.0076
20	0.0025	0.0025	0.0024
25	7.927e-04	7.887e-04	7.768e-04
30	2.498e-04	2.511e-04	2.458e-04

calculations of the BER and the distribution of the number of errors in a given block are performed using techniques described in appendix B.

The first comparison involves the analytical and predicted BER for several different SNRs, calculated using unscaled equation (4.22), in Table 10. The use of these error probabilities leads to a Markov model that overestimates the BER when using MMSE quantization. This pattern is not unique to this particular choice of fade rate and number of states (similar problem occurred with all other combinations) and serves to underscore the importance of weighting the error probability by the fade PDF. However, note that the the Markov model with logarithmic quantization intervals still provides fairly accurate results.

The analytical and predicted BER for several different SNRs, calculated using scaled equation (4.23), appear alongside each other in Table 11. One can see that both Markov models approximate the analytical BER very closely. This provides verification that the distribution of time spent in each state by the model matches theory.

The next step involves using the Markov models to calculate the number of errors in a given block of bits. Unlike the BER, no closed-form expression exists for the PDF of the number of errors in a block of bits. As a result, one must compare the distribution of the number of errors obtained from the Markov model to the results of waveform-level simulations. The results, for a SNR of 10, appear in Figure 26. One can see that the model with MMSE quantization intervals overestimates the number of low-weight patterns slightly and grossly underestimates the number of high-weight patterns. Again, this pattern is not unique to this particular choice of fade rate and number of states.

The model with logarithmic quantization levels performs much better. However, the block length of 25 corresponds to a time interval far less than the the coherence bandwidth of the channel.

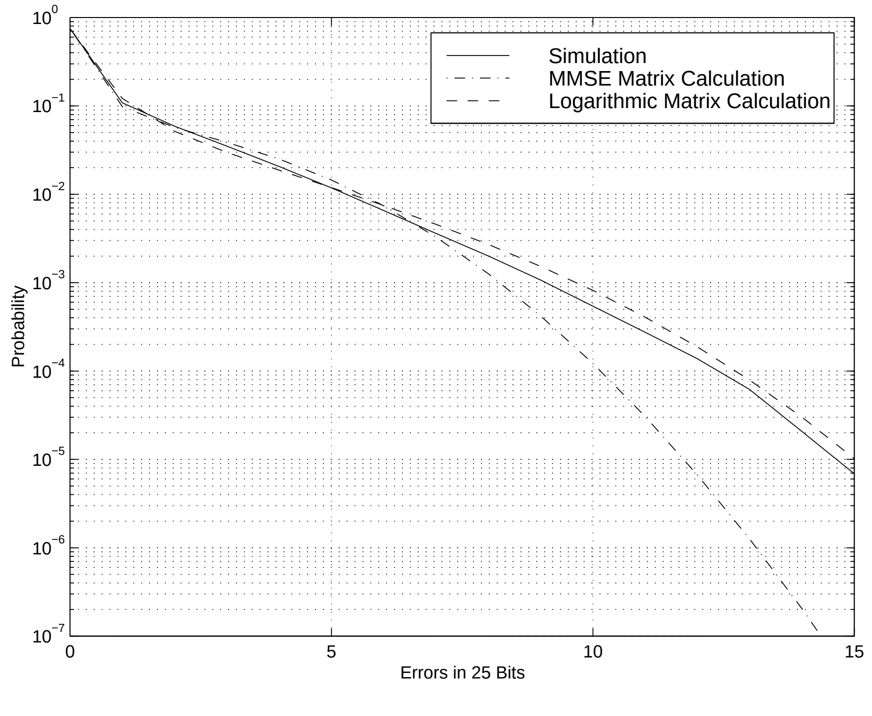

Figure 26: Error distribution, Rayleigh PDF, $FT_c = 0.01$, SNR=10dB, 25 bit blocks, Markov 8-Level quantization

Another example appears in Figure 27, which displays the results for a block length of 400 instead of 25. The block length exceeds the coherence bandwidth of the channel, so the model must take into account the long-term behavior of the channel. Based on Figure 27, one can see that the logarithmic Markov model performs fairly well and the MMSE Markov model performs rather poorly. In particular, examine the predicted values for the probability of zero errors.

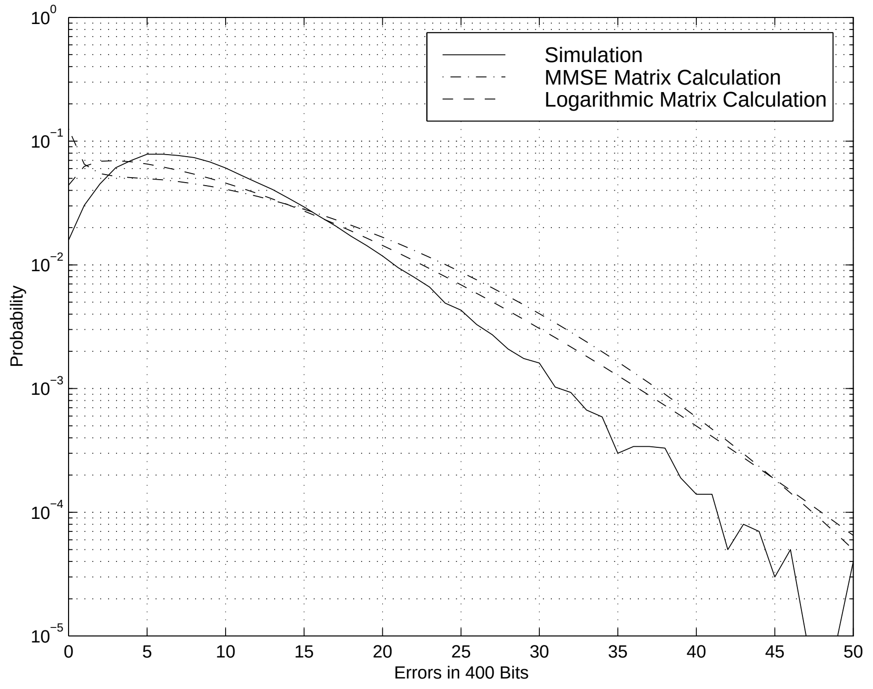

Figure 27: Error distribution, Rayleigh PDF, $FT_c = 0.01$, SNR=10dB, 400 bit blocks, Markov 8-Level quantization

Figure 28 presents a channel with a SNR of 0. Again, the curves show the error probabilities for blocks of 400 bits. The error distribution predicted by the logarithmic Markov model matches the simulation results fairly closely while the distribution predicted by the MMSE Markov model has a different shape (decreasing monotonically) from the simulated curve (concave down) and and the predicted values are nearly an order of magnitude off from the simulated values for small numbers of errors.

The deviations evident in the MMSE Markov model prohibit its use in the calculation of an optimal error control code, since the distribution and weight of the expected error patterns plays a major role in this decision.

Several papers have suggested modeling the fading process as a birth-death process [24] [22], which prohibits jumps to nonadjacent levels ($P_{i,j} = 0$ if $|i-j| > 1$) and makes possible the analytical calculation of state transition possibilities. The quantization levels for this model must be chosen to satisfy the nonadjacent level transition criteria. However, the states in a Markov model have exponentially-distributed dwell times, so the quantization levels

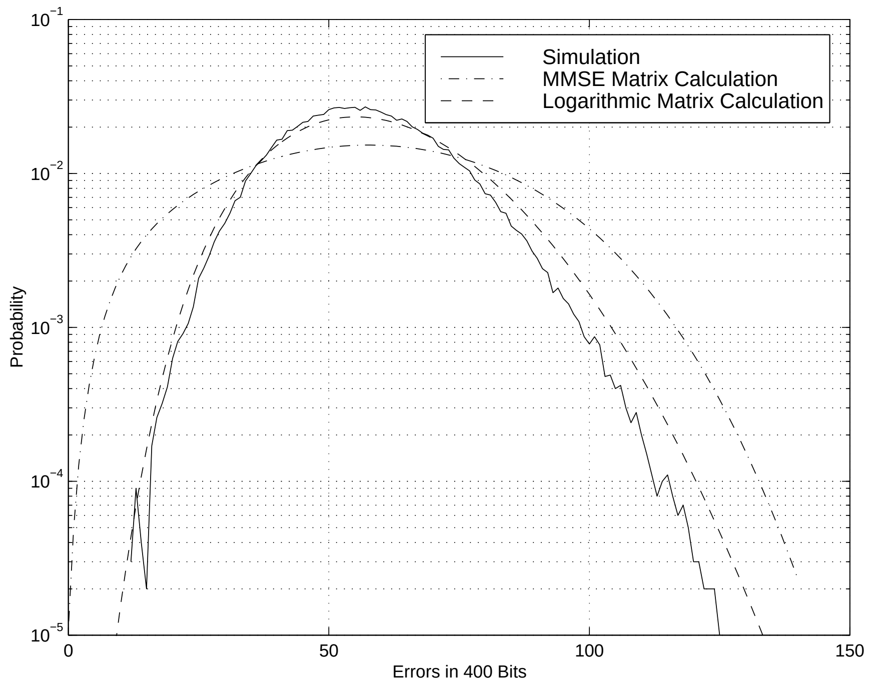

Figure 28: Error distribution, Rayleigh PDF, $FT_c = 0.01$, SNR=0dB, 400 bit blocks, HMM 8-Level quantization

must also be chosen to give exponentially-distributed state dwell times. The optimizations for these two factors run at cross purposes: increasing the number of states leads to state dwell times that tend towards exponential distributions, but increases the likelihood of nonadjacent state transitions. For this reason, this model is difficult to apply in practice and suffers from the same problems described above.

The fundamental problem with modeling state dwell times using a Markov model is illustrated in Figure 29. The PDF of the holding time of the fifth quantization interval (8-state logarithmic quantization) for a simulated Rayleigh fading process with $FT_c = 0.01$ appears in Figure 29 along with the same PDF for the Markov process fit to it. One can see that the actual PDF looks like a gamma distribution, while the Markov PDF describes an exponential distribution. Clearly, one cannot use an exponential distribution to model a gamma distribution with any degree of accuracy.

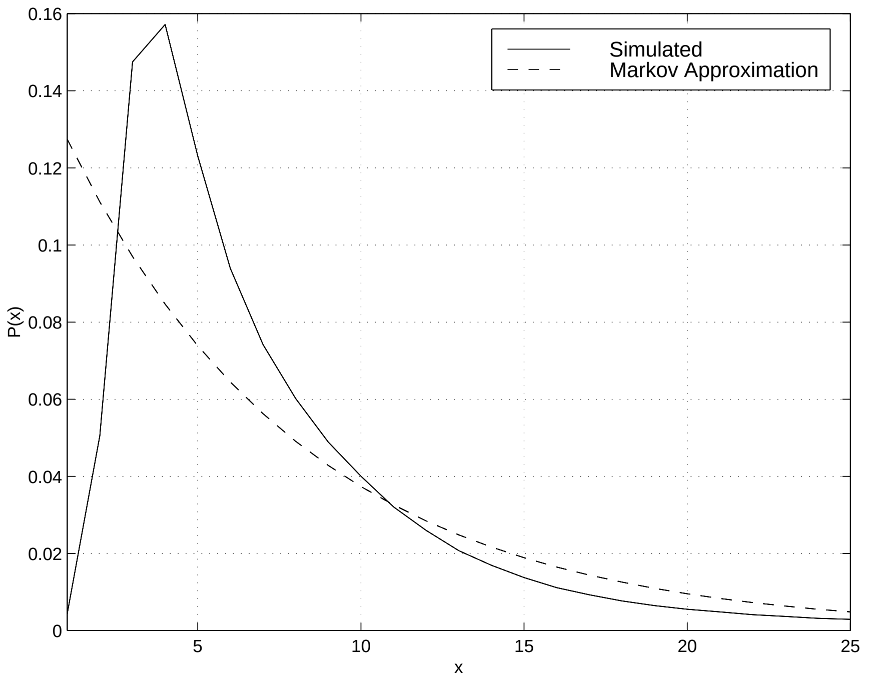

Figure 29: Simulated and Markov state holding times

4.5 Hidden Markov Models

A Markov model can be modified so that it will model a channel far more accurately. The process outlined for improving the model allows it to model the state holding times more closely. The modification involves fitting a phase-type (PH) distribution to the state holding time of each quantization interval. These distributions are then rolled into the state transition matrix to yield a hidden Markov model (HMM) matrix for the channel.

An example phase-type distribution appears in (4.26):

$$p(x) = \boldsymbol{\pi}\mathbf{A}^{x-1}\mathbf{b} \tag{4.26}$$

where $\boldsymbol{\pi}$ and $\mathbf{b}$ are vectors and $\mathbf{A}$ is a matrix such that the matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \boldsymbol{\pi} & 0 \end{bmatrix} \tag{4.27}$$

is a stochastic matrix, i.e., the rows sum to one. A detailed explanation the curve fitting techniques and some of the practical issues involved in calculating these matrices appears in Appendix C.

PDFs for the quantization intervals were obtained using the fading simulator written for GSM simulations. For each set of quantization intervals, 5,000,000 samples were created for each of 100 realizations of the fading process. These 100 distributions were then averaged together to mitigate the effects of parameter choice while keeping track of the standard deviation of each of the entries in the empirical PDF.

Given a holding time PDF for a quantization interval, one can calculate $\mathbf{A}, \boldsymbol{\pi}$ and $\mathbf{b}$ so that the model exhibits similar state holding time PDFs. An example appears in Figure 30, which shows the results of fitting a $5x5$ matrix $\mathbf{A}$ and $1x5$ and $5x1$ vectors $\boldsymbol{\pi}$ and $\mathbf{b}$ to the fifth logarithmic quantization interval of a simulated Rayleigh fading process with $FT_c = 0.01$. Note the improvement in accuracy over the exponential distribution shown in Figure 29.

A worst case example appears in Figure 31, which shows the fit for the seventh quantization interval. Again, the PH distribution parameters were fit to the empirical quantization

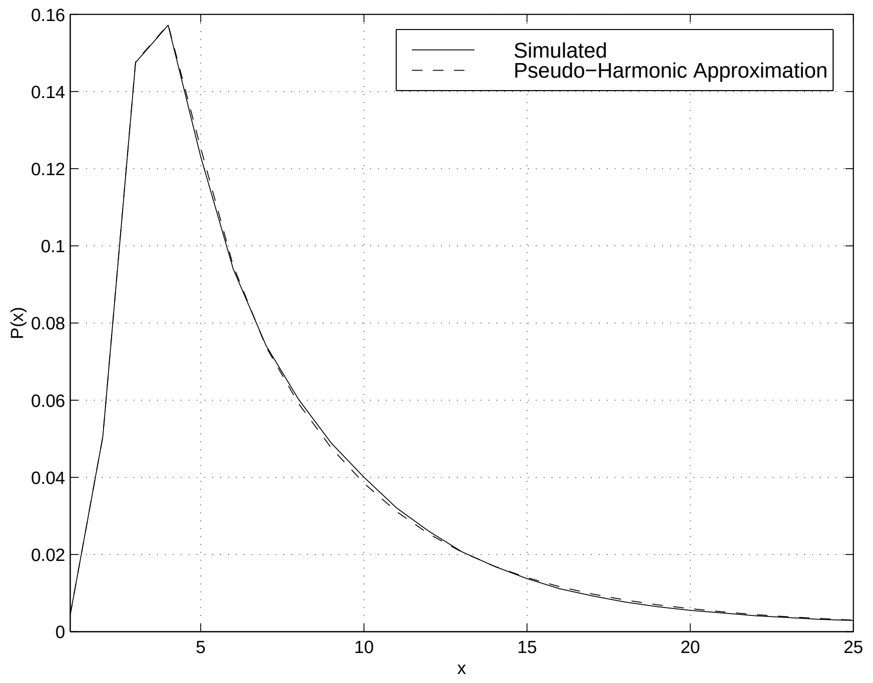

Figure 30: Simulated and hidden Markov model state holding times

interval PDF using a $5x5$ matrix $\mathbf{A}$ and $1x5$ and $5x1$ vectors $\boldsymbol{\pi}$ and $\mathbf{b}$. However, the bimodal state holding time PDF proved too complicated for the number of model parameters. Despite these difficulties, the PH distribution still models the empirical distribution far more accurately than the exponential distribution examined in the previous chapter.

The results of running the curve fitting algorithm with $10x10$ matrix $\mathbf{A}$ and $1x10$ and $10x1$ vectors $\boldsymbol{\pi}$ and $\mathbf{b}$ appear in Figure 32. The quality of the fit improved somewhat, but so did the execution time of the curve fitting algorithm increased substantially, since the number of independent parameters increased by a factor of four.

The next step was to create a large selection of models to analyze a a broad cross section of channels. For each fading channel, the PH distributions for each state (quantization interval) were combined to create the HMM for that channel (see chapter C). The models all used eight quantization levels, a $5x5$ matrix $\mathbf{A}$ and $1x5$ and $5x1$ vectors for $\boldsymbol{\pi}$ and $\mathbf{b}$, respectively. Quantization schemes for the models included: MMSE, the regions based on $\sqrt{2}/2$, those based on $\sqrt{\pi}/2$, and the regions from [1]. All of the logarithmic quantization schemes were tested with intervals of both 3 and 3.5dB. Models were generated for channel

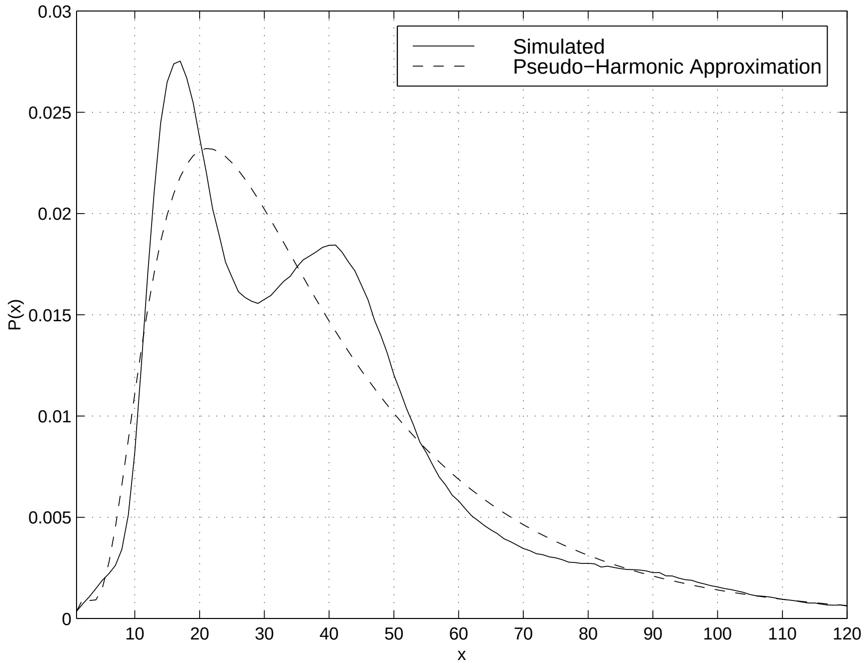

Figure 31: Simulated and hidden Markov model state holding times, 5x5 matrices

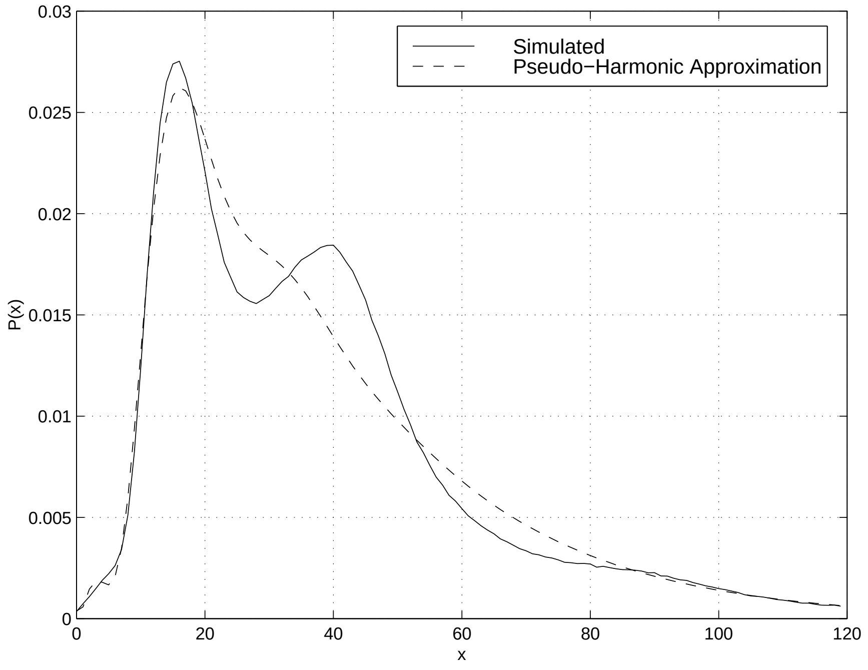

Figure 32: Simulated and hidden Markov model state holding times, 10x10 matrices

coherence bandwidths (FT_c) of $10^{-0.5}$=3.1623e-1, 10^{-1}=1e-1, $10^{-1.5}$=3.1623e-2, $10^{-2.0}$=1e-2, and $10^{-2.5}$=3.1623e-3. Tables 12 through 18 compare the analytical BER (using (4.20)) to the model BER (using (B.7)) for a range of SNRs between 0 and 30dB. The close match between the model and analytical values (less than 3% error in all cases, and generally within 1.5%) indicates a tight fit for the models.

Table 12: Analytical and model BERs, MMSE quantization

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.454e-01	1.452e-01	1.458e-01	1.459e-01	1.458e-01
2	1.085e-01	1.075e-01	1.072e-01	1.078e-01	1.078e-01	1.078e-01
4	7.714e-02	7.631e-02	7.606e-02	7.647e-02	7.652e-02	7.647e-02
6	5.300e-02	5.235e-02	5.214e-02	5.243e-02	5.247e-02	5.243e-02
8	3.546e-02	3.498e-02	3.482e-02	3.502e-02	3.505e-02	3.501e-02
10	2.327e-02	2.294e-02	2.282e-02	2.296e-02	2.297e-02	2.295e-02
12	1.506e-02	1.484e-02	1.477e-02	1.485e-02	1.486e-02	1.484e-02
14	9.665e-03	9.520e-03	9.470e-03	9.525e-03	9.531e-03	9.518e-03
16	6.164e-03	6.071e-03	6.038e-03	6.074e-03	6.078e-03	6.069e-03
18	3.916e-03	3.857e-03	3.836e-03	3.858e-03	3.861e-03	3.856e-03
20	2.481e-03	2.444e-03	2.431e-03	2.445e-03	2.447e-03	2.443e-03
22	1.570e-03	1.546e-03	1.538e-03	1.547e-03	1.548e-03	1.546e-03
24	9.923e-04	9.773e-04	9.721e-04	9.778e-04	9.784e-04	9.771e-04
26	6.268e-04	6.173e-04	6.140e-04	6.176e-04	6.180e-04	6.172e-04
28	3.958e-04	3.898e-04	3.877e-04	3.900e-04	3.902e-04	3.897e-04
30	2.498e-04	2.460e-04	2.447e-04	2.462e-04	2.463e-04	2.460e-04

These models were then used to calculate the number of errors in blocks of bits corresponding to the length of several error control codes: a 127-bit BCH code, a 31-symbol Reed-Solomon code, and a 255-symbol Reed Solomon code. These error distributions were then compared to waveform simulations. One of the interesting trends is that the results using the MMSE quantization regions became less accurate as the fade rate decreased: compare figures 33 and 34, both of which suggest very good fits, with figures 35 and 36, which are very poor fits. This pattern appeared across all of the plots created using the MMSE quantization scheme: the plots for the rapidly-changing channels matched the simulation results very closely, but the discrepancy between simulation and analytical results increased as the channel fade rate decreased.

Table 13: Analytical and model BERs, $\sqrt{2}/2$ quantization, 3.0dB spacing

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.451e-01	1.445e-01	1.461e-01	1.454e-01	1.464e-01
2	1.085e-01	1.072e-01	1.067e-01	1.080e-01	1.075e-01	1.084e-01
4	7.714e-02	7.602e-02	7.555e-02	7.662e-02	7.626e-02	7.693e-02
6	5.300e-02	5.211e-02	5.172e-02	5.252e-02	5.228e-02	5.277e-02
8	3.546e-02	3.479e-02	3.449e-02	3.506e-02	3.491e-02	3.525e-02
10	2.327e-02	2.280e-02	2.257e-02	2.296e-02	2.287e-02	2.311e-02
12	1.506e-02	1.474e-02	1.458e-02	1.484e-02	1.479e-02	1.494e-02
14	9.665e-03	9.451e-03	9.341e-03	9.508e-03	9.475e-03	9.580e-03
16	6.164e-03	6.024e-03	5.951e-03	6.057e-03	6.037e-03	6.107e-03
18	3.916e-03	3.825e-03	3.777e-03	3.845e-03	3.831e-03	3.878e-03
20	2.481e-03	2.423e-03	2.392e-03	2.435e-03	2.425e-03	2.457e-03
22	1.570e-03	1.533e-03	1.513e-03	1.540e-03	1.533e-03	1.555e-03
24	9.923e-04	9.685e-04	9.555e-04	9.732e-04	9.678e-04	9.824e-04
26	6.268e-04	6.116e-04	6.032e-04	6.147e-04	6.106e-04	6.204e-04
28	3.958e-04	3.860e-04	3.806e-04	3.881e-04	3.851e-04	3.916e-04
30	2.498e-04	2.435e-04	2.401e-04	2.450e-04	2.428e-04	2.472e-04

Table 14: Analytical and model BERs, $\sqrt{2}/2$ quantization, 3.5dB spacing

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.474e-01	1.460e-01	1.472e-01	1.467e-01	1.463e-01
2	1.085e-01	1.092e-01	1.079e-01	1.090e-01	1.085e-01	1.083e-01
4	7.714e-02	7.773e-02	7.661e-02	7.745e-02	7.707e-02	7.688e-02
6	5.300e-02	5.344e-02	5.253e-02	5.316e-02	5.288e-02	5.274e-02
8	3.546e-02	3.576e-02	3.508e-02	3.552e-02	3.533e-02	3.524e-02
10	2.327e-02	2.347e-02	2.298e-02	2.328e-02	2.315e-02	2.309e-02
12	1.506e-02	1.520e-02	1.486e-02	1.505e-02	1.497e-02	1.493e-02
14	9.665e-03	9.751e-03	9.523e-03	9.642e-03	9.594e-03	9.574e-03
16	6.164e-03	6.219e-03	6.069e-03	6.142e-03	6.113e-03	6.103e-03
18	3.916e-03	3.951e-03	3.854e-03	3.898e-03	3.880e-03	3.876e-03
20	2.481e-03	2.504e-03	2.441e-03	2.468e-03	2.456e-03	2.456e-03
22	1.570e-03	1.584e-03	1.544e-03	1.561e-03	1.552e-03	1.553e-03
24	9.923e-04	1.001e-03	9.755e-04	9.859e-04	9.802e-04	9.813e-04
26	6.268e-04	6.324e-04	6.159e-04	6.224e-04	6.186e-04	6.195e-04
28	3.958e-04	3.993e-04	3.887e-04	3.928e-04	3.903e-04	3.908e-04
30	2.498e-04	2.520e-04	2.453e-04	2.478e-04	2.463e-04	2.465e-04

Table 15: Analytical and model BERs, $\sqrt{\pi}/2$ quantization, 3.0dB spacing

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.458e-01	1.456e-01	1.442e-01	1.467e-01	1.469e-01
2	1.085e-01	1.078e-01	1.076e-01	1.063e-01	1.086e-01	1.088e-01
4	7.714e-02	7.652e-02	7.630e-02	7.528e-02	7.718e-02	7.727e-02
6	5.300e-02	5.249e-02	5.229e-02	5.151e-02	5.298e-02	5.303e-02
8	3.546e-02	3.507e-02	3.489e-02	3.433e-02	3.541e-02	3.544e-02
10	2.327e-02	2.298e-02	2.285e-02	2.245e-02	2.322e-02	2.323e-02
12	1.506e-02	1.487e-02	1.477e-02	1.450e-02	1.502e-02	1.502e-02
14	9.665e-03	9.534e-03	9.465e-03	9.282e-03	9.630e-03	9.630e-03
16	6.164e-03	6.078e-03	6.031e-03	5.910e-03	6.137e-03	6.137e-03
18	3.916e-03	3.860e-03	3.829e-03	3.750e-03	3.896e-03	3.896e-03
20	2.481e-03	2.446e-03	2.425e-03	2.374e-03	2.467e-03	2.468e-03
22	1.570e-03	1.547e-03	1.534e-03	1.501e-03	1.560e-03	1.561e-03
24	9.923e-04	9.780e-04	9.692e-04	9.486e-04	9.849e-04	9.863e-04
26	6.268e-04	6.178e-04	6.120e-04	5.991e-04	6.217e-04	6.228e-04
28	3.958e-04	3.901e-04	3.863e-04	3.782e-04	3.923e-04	3.931e-04
30	2.498e-04	2.463e-04	2.439e-04	2.388e-04	2.475e-04	2.480e-04

Table 16: Analytical and model BERs, $\sqrt{\pi}/2$ quantization, 3.5dB spacing

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.452e-01	1.459e-01	1.467e-01	1.463e-01	1.448e-01
2	1.085e-01	1.073e-01	1.079e-01	1.086e-01	1.082e-01	1.069e-01
4	7.714e-02	7.613e-02	7.659e-02	7.711e-02	7.685e-02	7.580e-02
6	5.300e-02	5.220e-02	5.251e-02	5.290e-02	5.272e-02	5.194e-02
8	3.546e-02	3.486e-02	3.506e-02	3.534e-02	3.522e-02	3.466e-02
10	2.327e-02	2.284e-02	2.296e-02	2.315e-02	2.307e-02	2.269e-02
12	1.506e-02	1.477e-02	1.484e-02	1.496e-02	1.492e-02	1.467e-02
14	9.665e-03	9.468e-03	9.511e-03	9.587e-03	9.560e-03	9.399e-03
16	6.164e-03	6.034e-03	6.061e-03	6.107e-03	6.091e-03	5.989e-03
18	3.916e-03	3.831e-03	3.848e-03	3.876e-03	3.865e-03	3.802e-03
20	2.481e-03	2.427e-03	2.438e-03	2.454e-03	2.447e-03	2.408e-03
22	1.570e-03	1.535e-03	1.542e-03	1.552e-03	1.546e-03	1.523e-03
24	9.923e-04	9.697e-04	9.745e-04	9.805e-04	9.763e-04	9.622e-04
26	6.268e-04	6.124e-04	6.155e-04	6.192e-04	6.162e-04	6.076e-04
28	3.958e-04	3.866e-04	3.887e-04	3.910e-04	3.888e-04	3.836e-04
30	2.498e-04	2.440e-04	2.454e-04	2.469e-04	2.453e-04	2.421e-04

Table 17: Analytical and model BERs, [1] quantization, 3.0dB spacing

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.460e-01	1.467e-01	1.452e-01	1.464e-01	1.452e-01
2	1.085e-01	1.080e-01	1.085e-01	1.073e-01	1.083e-01	1.073e-01
4	7.714e-02	7.664e-02	7.705e-02	7.603e-02	7.687e-02	7.605e-02
6	5.300e-02	5.257e-02	5.285e-02	5.208e-02	5.273e-02	5.211e-02
8	3.546e-02	3.512e-02	3.530e-02	3.474e-02	3.522e-02	3.478e-02
10	2.327e-02	2.302e-02	2.313e-02	2.274e-02	2.307e-02	2.278e-02
12	1.506e-02	1.489e-02	1.495e-02	1.469e-02	1.492e-02	1.473e-02
14	9.665e-03	9.547e-03	9.585e-03	9.408e-03	9.561e-03	9.440e-03
16	6.164e-03	6.086e-03	6.109e-03	5.992e-03	6.091e-03	6.017e-03
18	3.916e-03	3.866e-03	3.879e-03	3.802e-03	3.866e-03	3.821e-03
20	2.481e-03	2.449e-03	2.458e-03	2.408e-03	2.448e-03	2.421e-03
22	1.570e-03	1.550e-03	1.555e-03	1.523e-03	1.547e-03	1.532e-03
24	9.923e-04	9.792e-04	9.828e-04	9.622e-04	9.771e-04	9.679e-04
26	6.268e-04	6.183e-04	6.209e-04	6.077e-04	6.167e-04	6.112e-04
28	3.958e-04	3.903e-04	3.921e-04	3.838e-04	3.892e-04	3.858e-04
30	2.498e-04	2.463e-04	2.476e-04	2.423e-04	2.455e-04	2.435e-04

Table 18: Analytical and model BERs, [1] quantization, 3.5dB spacing

SNR	analytical	FT_c=0.32	FT_c=0.1	FT_c=3.2e-2	FT_c=1e-2	FT_c=3.2e-3
0	1.464e-01	1.474e-01	1.476e-01	1.479e-01	1.479e-01	1.470e-01
2	1.085e-01	1.092e-01	1.093e-01	1.096e-01	1.096e-01	1.088e-01
4	7.714e-02	7.770e-02	7.773e-02	7.798e-02	7.798e-02	7.734e-02
6	5.300e-02	5.340e-02	5.338e-02	5.357e-02	5.357e-02	5.310e-02
8	3.546e-02	3.572e-02	3.568e-02	3.582e-02	3.582e-02	3.550e-02
10	2.327e-02	2.344e-02	2.339e-02	2.348e-02	2.348e-02	2.327e-02
12	1.506e-02	1.517e-02	1.513e-02	1.518e-02	1.518e-02	1.505e-02
14	9.665e-03	9.731e-03	9.697e-03	9.731e-03	9.731e-03	9.652e-03
16	6.164e-03	6.204e-03	6.181e-03	6.199e-03	6.199e-03	6.153e-03
18	3.916e-03	3.941e-03	3.925e-03	3.934e-03	3.934e-03	3.906e-03
20	2.481e-03	2.497e-03	2.486e-03	2.491e-03	2.491e-03	2.474e-03
22	1.570e-03	1.580e-03	1.573e-03	1.575e-03	1.575e-03	1.564e-03
24	9.923e-04	9.984e-04	9.936e-04	9.950e-04	9.950e-04	9.878e-04
26	6.268e-04	6.308e-04	6.275e-04	6.283e-04	6.283e-04	6.234e-04
28	3.958e-04	3.984e-04	3.961e-04	3.966e-04	3.966e-04	3.933e-04
30	2.498e-04	2.517e-04	2.500e-04	2.504e-04	2.504e-04	2.481e-04

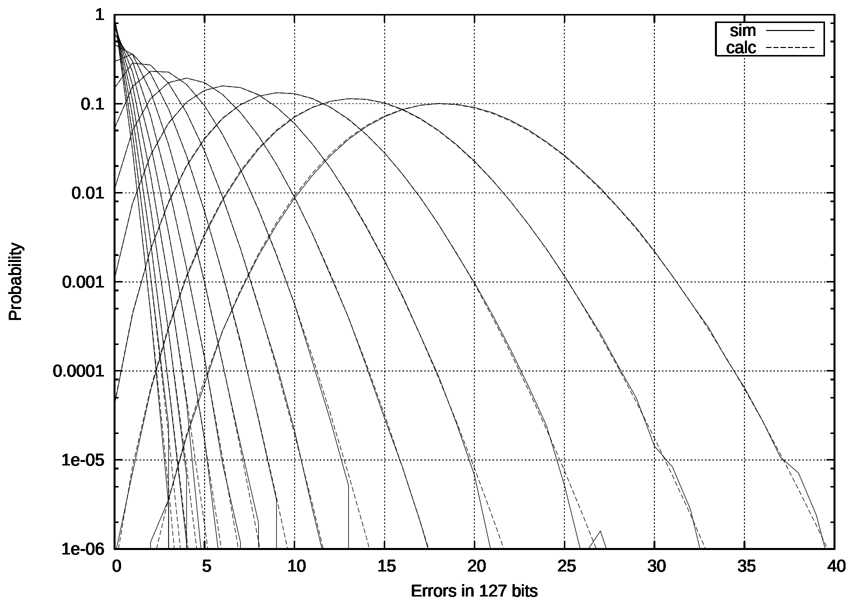

Figure 33: Error distribution, Rayleigh PDF, FT_c=.32, SNR=0-30dB, 127 bit blocks, MMSE 8-Level quantization

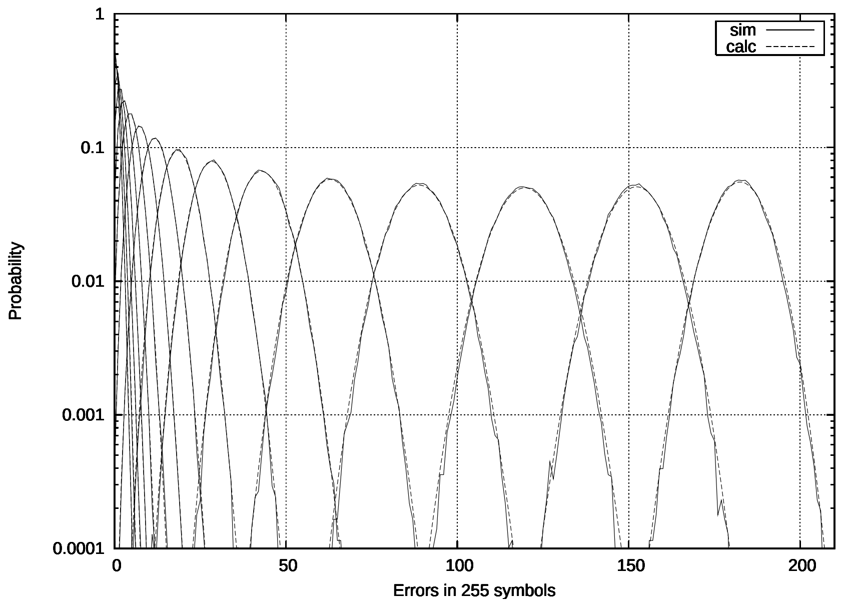

Figure 34: Error distribution, Rayleigh PDF, FT_c=.32, SNR=0-30dB, 255 symbol blocks, MMSE 8-Level quantization

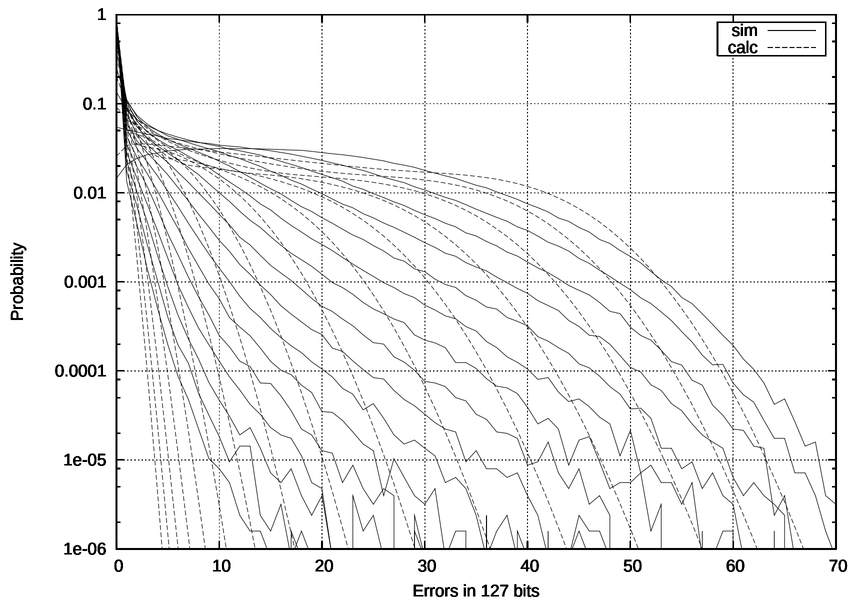

Figure 35: Error distribution, Rayleigh PDF, FT_c=3.2e-3, SNR=0-30dB, 127 bit blocks, MMSE 8-Level quantization

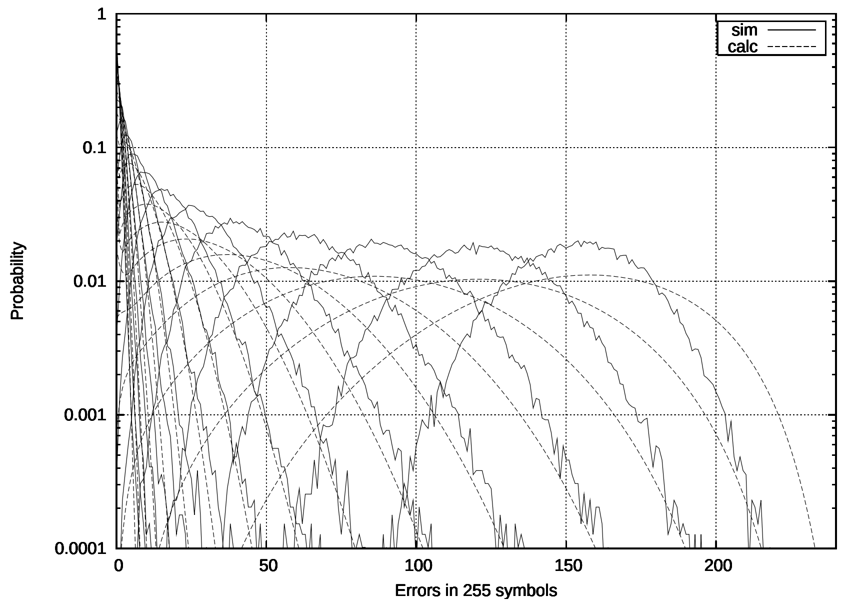

Figure 36: Error distribution, Rayleigh PDF, FT_c=3.2e-3, SNR=0-30dB, 255 symbol blocks, MMSE 8-Level quantization

The plots made using the logarithmic schemes shows a much better fit across a broad range of channels. Figures 37 and 38, like MMSE figures 33 and 34, indicate very good fits. However, figures 39 and 40, which show plots for more slowly-varying channels, also demonstrate close matches between the matrix calculations and simulations. This pattern remained consistent across all of the different SNRs and block lengths investigated.

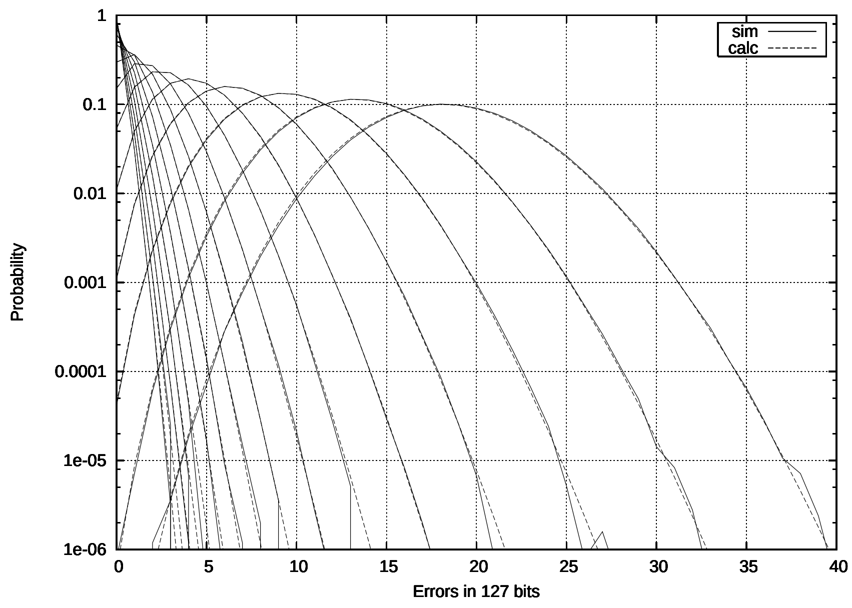

Figure 37: Error distribution, Rayleigh PDF, FT_c=.32, SNR=0-30dB, 127 bit blocks, $\sqrt{\pi}/2$ 8-Level quantization, 3.5dB intervals

The precise choice of quantization regions had little effect on the agreement between simulation results and the matrix calculations. Figures 41 through 46 show the results of six different quantization schemes while figure 47 provides a MMSE reference. One of the logarithmic schemes may appear slightly more accurate than another, but no general pattern held across all of the cases.

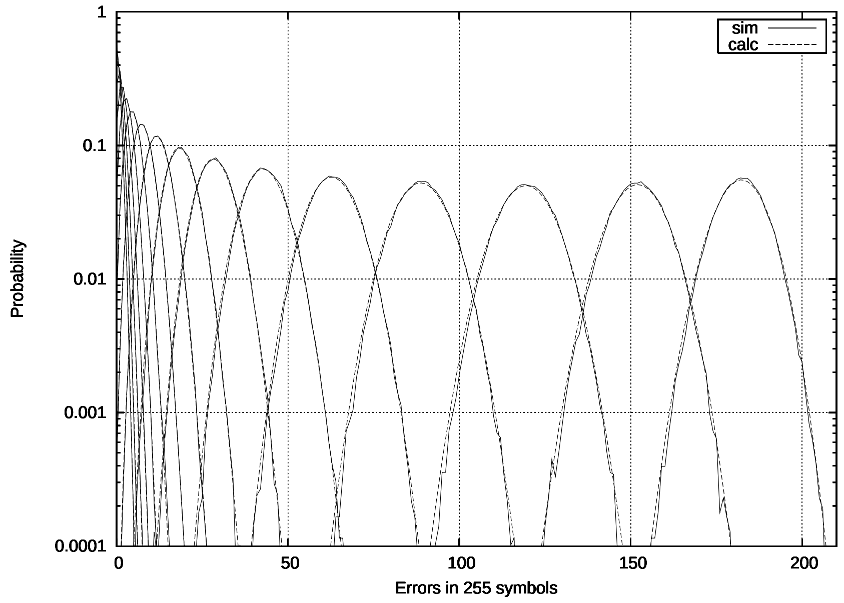

Figure 38: Error distribution, Rayleigh PDF, FT_c=.32, SNR=0-30dB, 255 symbol blocks, $\sqrt{\pi}/2$ 8-Level quantization, 3.5dB intervals

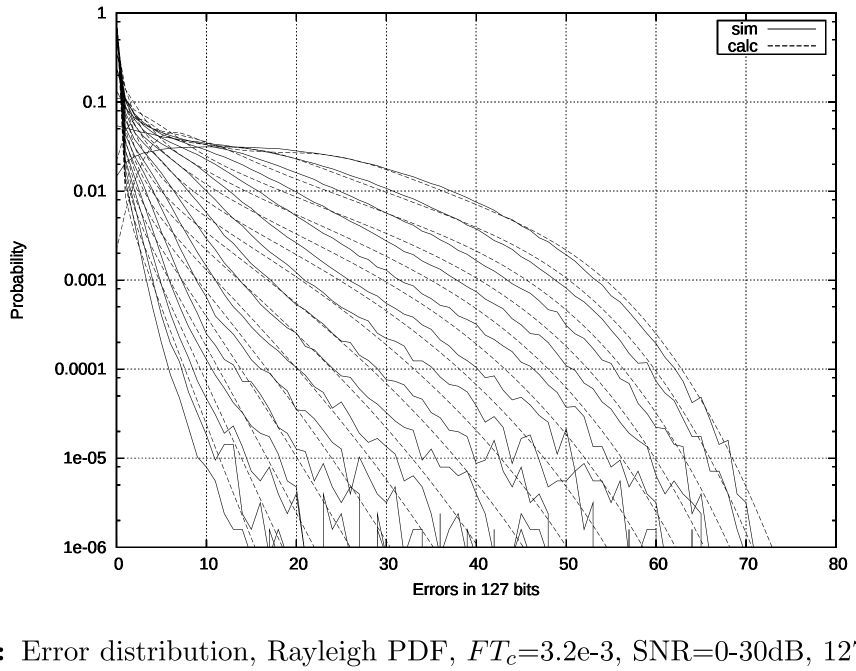

Figure 39: Error distribution, Rayleigh PDF, FT_c=3.2e-3, SNR=0-30dB, 127 bit blocks, $\sqrt{\pi}/2$ 8-Level quantization, 3.5dB intervals

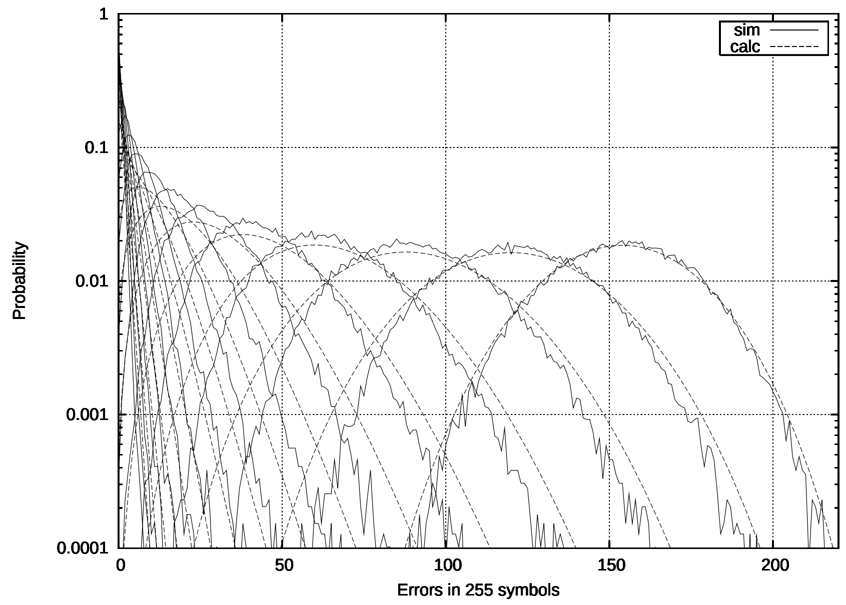

Figure 40: Error distribution, Rayleigh PDF, FT_c=3.2e-3, SNR=0-30dB, 255 symbol blocks, $\sqrt{\pi}/2$ 8-Level quantization, 3.5dB intervals

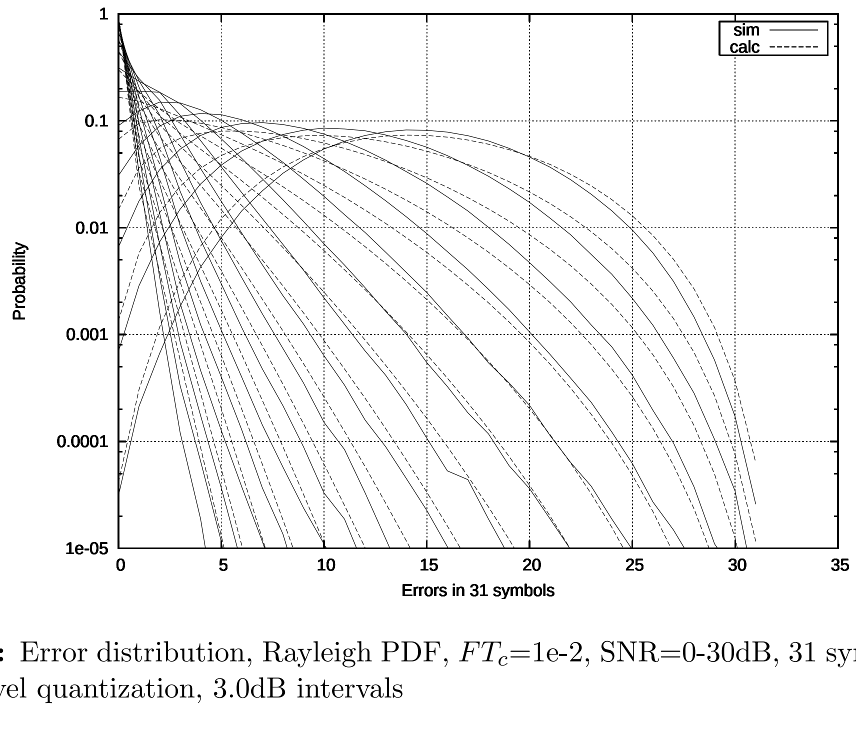

Figure 41: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, $\sqrt{2}/2$ 8-Level quantization, 3.0dB intervals

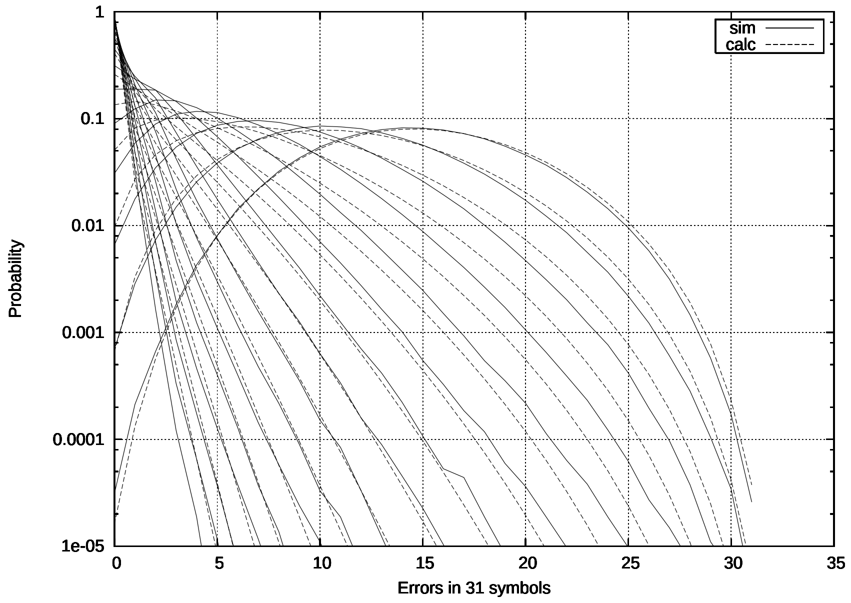

Figure 42: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, $\sqrt{2}/2$ 8-Level quantization, 3.5dB intervals

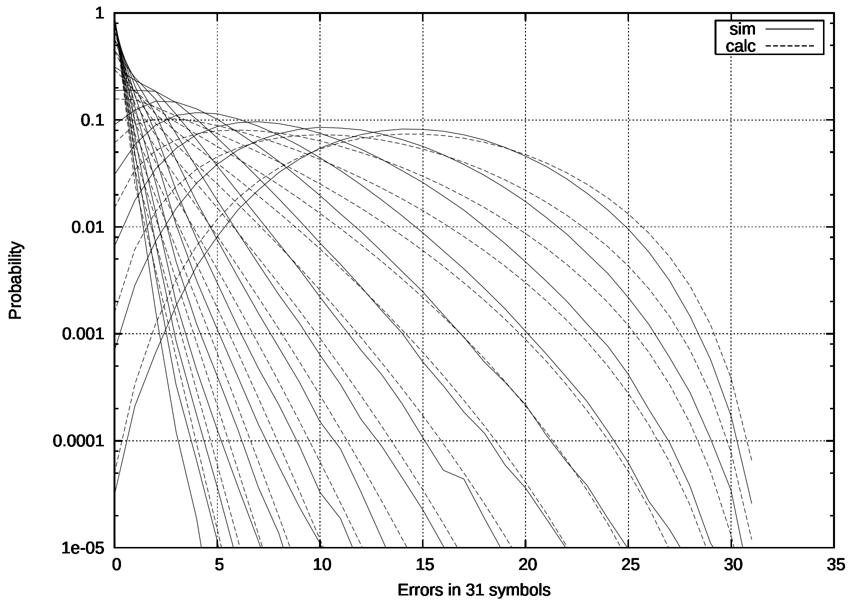

Figure 43: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, $\sqrt{\pi}/2$ 8-Level quantization, 3.0dB intervals

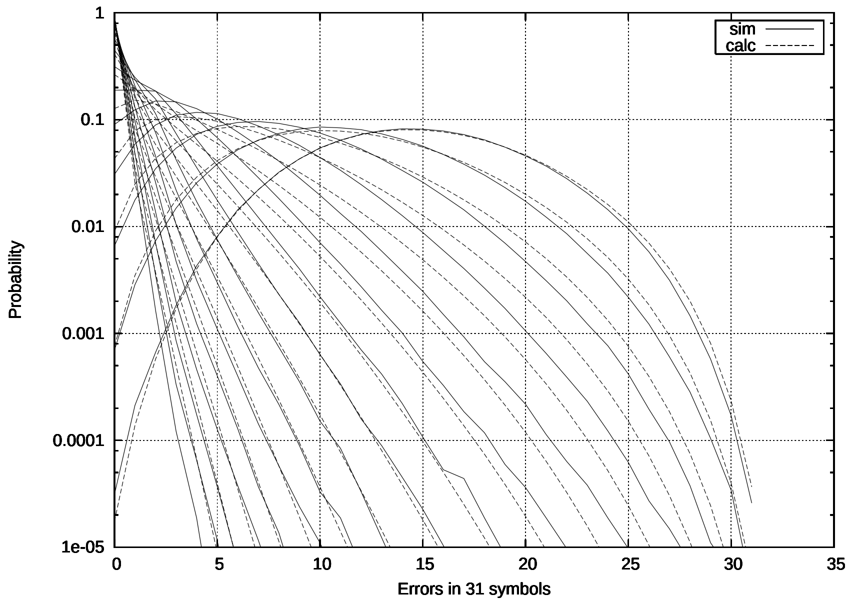

Figure 44: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, $\sqrt{\pi}/2$ 8-Level quantization, 3.5dB intervals

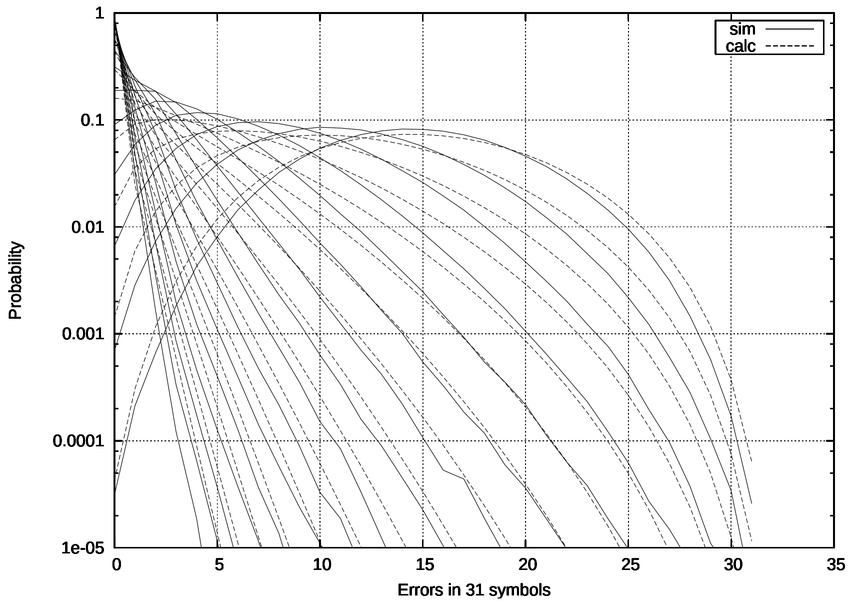

Figure 45: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, [1] 8-Level quantization, 3.0dB intervals

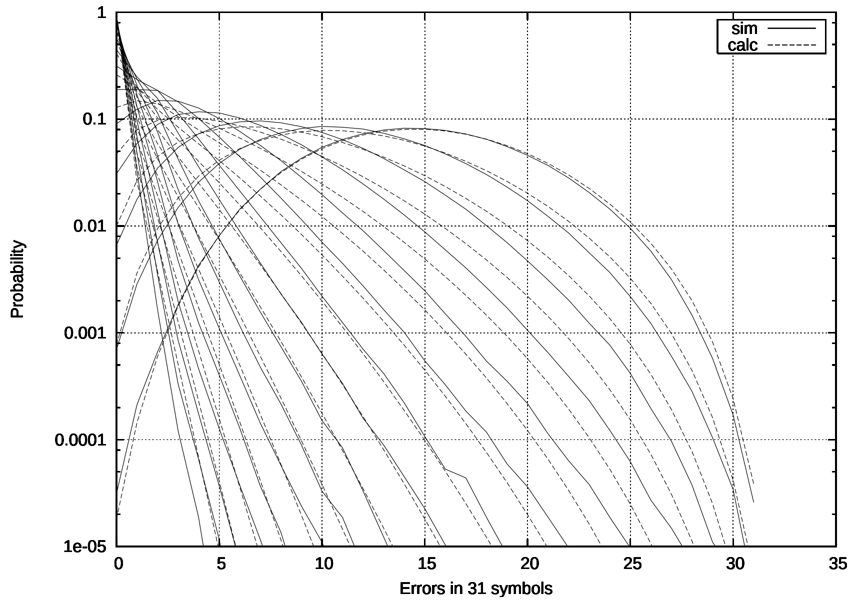

Figure 46: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, [1] 8-Level quantization, 3.5dB intervals

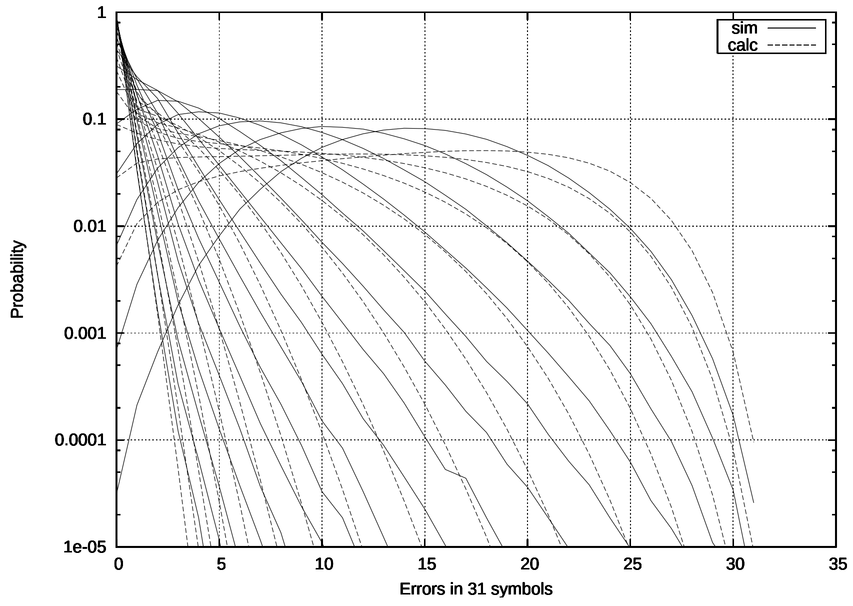

Figure 47: Error distribution, Rayleigh PDF, FT_c=1e-2, SNR=0-30dB, 31 symbol blocks, MMSE 8-Level quantization, 3.0dB intervals

4.6 Optimization

Choosing an error control code for a particular wireless application presents a difficult problem.

Traditional transport protocols, such as those in the the TCP/IP protocol suite and ATM, were designed for use across relatively error-free channels, and often run into problems when faced with a channel that introduces a significant number of errors. TCP, for instance, treats packet loss as network congestion and and throttles back packet transmission.

Wireless channels have a much higher BER, lower bandwidth, and often vary in time, factors that bring out the worst in higher-layer protocols. These problems, however, can be addressed through clever MAC protocols and error control schemes.

One may have different goals which will depend on the traffic's QoS requirements:

- Maximize the throughput subject to BER, delay, and jitter requirements.
- Minimize the BER subject to throughput, delay, and jitter requirements.
- Minimize the delay subject to throughput, jitter, and BER requirements.
- Minimize the jitter subject to throughput, delay, and BER requirements.
- Some other combination of QoS parameters

4.6.1 AWGN Example

This example will involve an AWGN channel and investigate the first case (maximize throughbut subject to BER, delay, and jitter requirements), which would occur when transmitting a video or audio signal. The example will use Reed-Solomon codes as the example error control code. The use of the HMM channels developed in previous chapters will come later.

The equation that relates the SNR of an AWGN channel to the received BER when using BPSK modulation appears in (4.13). This probability of error (p) may be viewed as the toss of a weighted coin, where the bit will be received in error with probability p. For nonbinary codes over GF(2^m), the symbol error probability, which will be called γ, equals the probability of m consecutive successful transmissions when using BPSK: $\gamma = 1-(1-p)^m$.

For codes over GF(2), one has the case where $\gamma = p$. The symbol p will be used for the symbol error probability to avoid introducing more symbols than are necessary.

The mean and standard deviation of n binomial trials are $\mu = np$ and $\sigma^2 = np(1-p)$, respectively [37]. Thus, for an (n, k) code, the number of errors in each received vector will have $\mu = n\gamma$ and $\sigma^2 = n\gamma(1-\gamma)$.

One can then calculate the probability of m errors in n received symbols using the binomial law:

$$P(m) = \binom{n}{m} p^m (1-p)^{n-m} \tag{4.28}$$

If $n \gg 1$ and $p \ll 1$ but their product (the distribution mean) np remains finite, one can approximate (4.28) using the Poisson law: [38]

$$P(m) \simeq \frac{(np)^k}{k!} e^{-np} \tag{4.29}$$

One can further approximate (4.29) using normal PDFs if $np \gg 1$: [38]

$$P(m) \simeq \frac{1}{\sqrt{2\pi}} \int_{l_1}^{l_2} e^{-\frac{1}{2}x^2} dx \tag{4.30}$$

where

$$l_1 = \frac{m - np - 0.5}{\sqrt{np(1-p)}}, \quad l_2 = \frac{m - np + 0.5}{\sqrt{np(1-p)}}$$

One can then rewrite (4.30) in terms of the error function: [4]

$$P(m) \simeq \frac{1}{2}\left(\operatorname{erf}\left(\frac{l_2}{\sqrt{2}}\right) - \operatorname{erf}\left(\frac{l_1}{\sqrt{2}}\right)\right) \tag{4.31}$$

where l_1 and l_2 are as above.

An example approximation of the binomial distribution using the error function (4.31) appears in Figure 48.

[4] $\operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-\lambda^2} d\lambda$

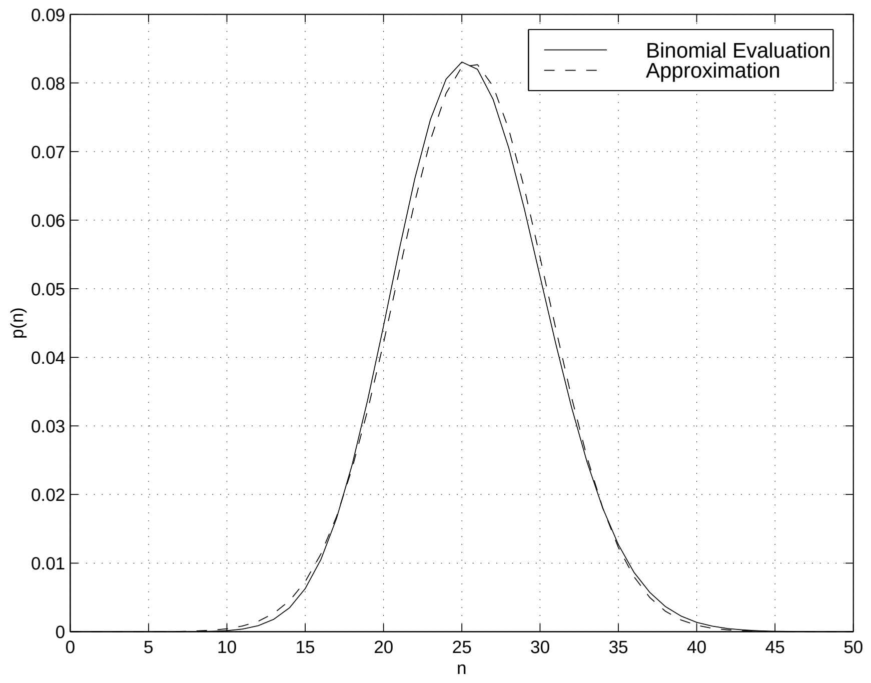

Figure 48: Example error function approximation to the binomial distribution, $n = 256, p = 0.1$

A vector will be received in error if it contains more errors than the code can correct. The probability of error for a t-error correcting code is then the probability of receiving more than t symbols in error:

$$P(\text{error}) = \sum_{m=t+1}^{n} \binom{n}{m} p^m (1-p)^{n-m} \tag{4.32}$$

As with the binomial PDF in (4.28), one can perform a series of approximations that allow us to express this result in terms of the error function. The Poisson approximation from (4.29) allows for slightly easier evaluation of (4.32):

$$P(\text{error}) \simeq \sum_{m=t+1}^{n} \frac{(np)^k}{k!} e^{-np} \tag{4.33}$$

The normal PDF approximation from (4.30) gives:

$$P(\text{error}) \simeq \sum_{m=t+1}^{n} \frac{(np)^k}{k!} e^{-np} \simeq \frac{1}{\sqrt{2\pi}} \int_{l_1}^{l_2} e^{-\frac{1}{2}x^2} dx \tag{4.34}$$

where

$$l_1 = \frac{(t+1) - np - 0.5}{\sqrt{np(1-p)}} = \frac{t - np + 0.5}{\sqrt{np(1-p)}}, \quad l_2 = \frac{(n) - np + 0.5}{\sqrt{np(1-p)}}$$

One can then rewrite (4.34) in terms of the error function:

$$\begin{aligned} P(\text{error}) &\simeq \tfrac{1}{2}\left(\text{erf}\left(\tfrac{l_2}{\sqrt{2}}\right) - \text{erf}\left(\tfrac{l_1}{\sqrt{2}}\right)\right) \\ &\simeq \tfrac{1}{2}\,\text{erfc}\left(\tfrac{l_1}{\sqrt{2}}\right) \\ &= \tfrac{1}{2}\,\text{erfc}\left(\tfrac{t-np+0.5}{\sqrt{2np(1-p)}}\right) \end{aligned} \tag{4.35}$$

Hence,

$$\begin{aligned} P(\text{success}) &= 1 - P(\text{error}) \\ P(\text{success}) &\simeq 1 - \tfrac{1}{2}\,\text{erfc}\left(\tfrac{l_1}{\sqrt{2}}\right) \end{aligned} \tag{4.36}$$

The throughput is the code rate times the success rate. A t error correcting Reed-Solomon code has a rate of $(n-2t)/n$, which gives:

$$\begin{aligned} \text{throughput} &= \tfrac{n-2t}{n}\left(1 - \tfrac{1}{2}\,\text{erfc}\left(\tfrac{l_1}{\sqrt{2}}\right)\right) \\ &= \tfrac{n-2t}{n}\left(1 - \tfrac{1}{2}\,\text{erfc}\left(\tfrac{t-np+0.5}{\sqrt{2np(1-p)}}\right)\right) \end{aligned} \tag{4.37}$$

Which one can differentiate (4.37) and solve for the value of t that causes the equation to evaluate to zero to find the maximum:

$$\frac{2}{n}\left(\frac{n-2t}{2\sqrt{2np(1-p)\pi}}e^{-\left(\frac{t-np+0.5}{\sqrt{2np(1-p)}}\right)^2} + \text{erfc}\left(\frac{t-np+0.5}{\sqrt{2np(1-p)}}\right) - 1\right) = 0 \tag{4.38}$$

Naturally, the value of t that brings this about will not necessarily be an integer. Thus, it is necessary to check the throughput brought about by an error control code with error correcting ability equal to the integers on either side of the computed value.

Solving (4.38) numerically for the example values gives 34.91. Since one cannot devise a code than corrects 34.91 errors, it is necessary to calculate the throughput for $t = 34$ and $t = 35$ and see which is greater. The throughput for $t = 35$ (0.7123) exceeds that for $t = 34$ (0.7110), so one should therefore choose $t = 35$ to obtain to maximum throughput for this example.

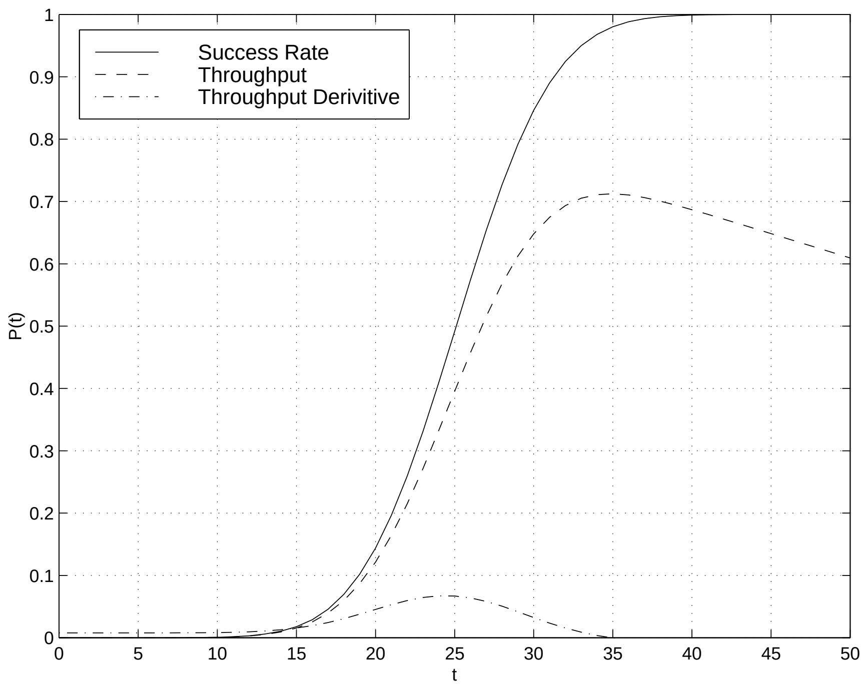

Figure 49: Example throughput curves, $n = 256, p = 0.1$

Plots of the success rate (4.36), throughput (4.37), and the derivative of the throughput (4.38) appear in Figure 49. Closer inspection of Figure 49 reveals that maximizing throughput results in receiving some packets in error. In this example, approximately five out of every 200 packets will contain too many errors for the code to correct, and higher layers in the communications stack (e.g., IP) will most likely drop these packets if the Reed-Solomon decoder does not. Whether or not such a rate of packet loss will prove unacceptable will depend upon the application. One would use a more powerful error correcting code if the rate of packet loss would prove unacceptable, or use a different data format. For example, full-motion JPEG video is far more tolerant of errors than MPEG video since there is no dependence between frames.

One can calculate an upper bound on the residual bit error rate using a well-known property of Reed-Solomon codes: an (n, k) Reed-Solomon code will always correct vectors with $t = (n-k)/2$ or fewer errors and will signal a decoder failure in received vectors having more than t errors with probability: [39]

$$P(\text{decoder error}) = \frac{1}{t!} \tag{4.39}$$

Thus, one can expect that one out of every $t!$ erroneous vectors will result in a decoder failure, i.e., the erroneous vector lies within the decoding sphere of a codeword and will fool the decoder. For the purposes of upper bounding the bit error rate, one can assume that the every bit in the incorrect vector is erroneous. Table 19 shows the effect of varying t on the code rate, the probability of decoder error, and the expected number erroneous vectors correctly declared decoder failures between each decoder error.

Table 19: Decoder error probabilities for a Reed-Solomon code over GF(256)

R-S Code	t	Code Rate	P(decoder error)	Error Period
(255,251)	2	.9843	0.5	2
(255,245)	5	.9608	8.3333e-03	120
(255,235)	10	.9216	2.7557e-07	3,628,800
(255,205)	25	.8039	6.4470e-26	1.5511e+25

Multiplying together the probability of decoder failure (4.39), the probability of receiving a vector with more than t errors (4.35) gives an upper bound on the BER resulting from decoder failures:

$$P(\text{bit error}) \leq \frac{1}{2t!}\,\text{erfc}\left(\frac{t - np + 0.5}{\sqrt{2np(1-p)}}\right) \tag{4.40}$$

This graph of this equation for the example system is juxtaposed with much of the information from Figure 49 in Figure 50. One can see that the residual BER decreases as t increases, both before and after the peak throughput. The peak throughput point of $t = 35$ corresponds to a residual BER of 1.8949e-42, which should prove sufficient for any realistic application.

4.6.2 Fading Channel

The analysis of codes for a fading channel poses difficulties that do not exist in the AWGN case. These difficulties are due to the correlation between symbols inherent in the fading

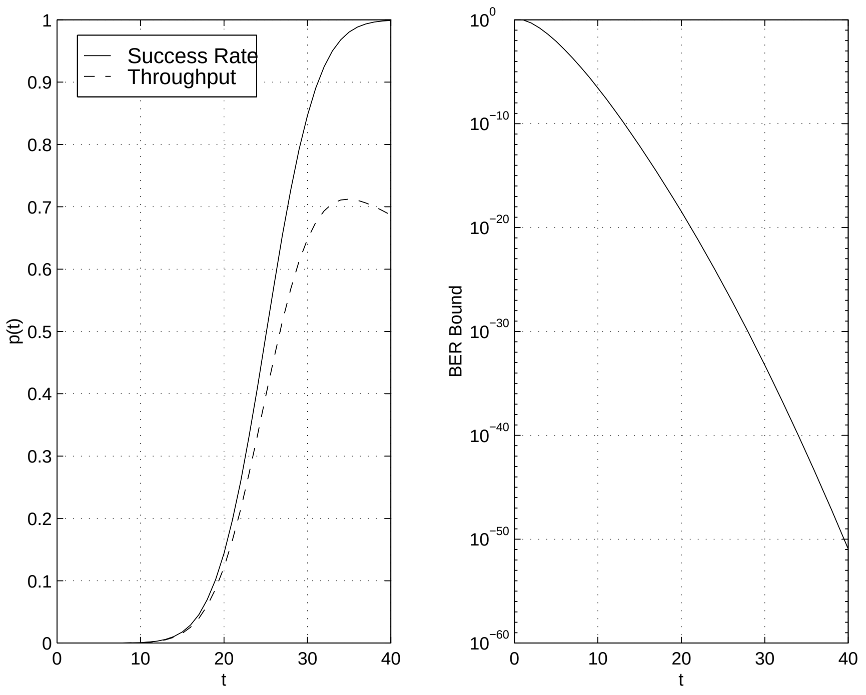

Figure 50: Left: success rate and throughput, Right: BER upper bound

channel. Hence, the simple binomial approximation breaks down and generally cannot be used[5].

Instead, one use the channel models developed earlier to estimate the probability of various error patterns. One can then evaluate different codes and find one appropriate to the design goal. For example, going back to the Rayleigh fading channel with $FT_c = 0.01$, consider the case of length 127 binary BCH codes. This family of codes gives us the options shown in Table 20.

Table 20: Length 127 BCH codes generated by Matlab's default primitive polynomial.

K	T	K	T	K	T	K	T
120	1	85	6	50	13	15	27
113	2	78	7	43	14	8	31
106	3	71	9	36	15		
99	4	64	10	29	21		
92	5	57	11	22	23		

[5]Note that the binomial approximation may be used if the channel fade rate is much less than the bit/frame length.

One can pick an SNR and quickly calculate the performance of each code using the following techniques.

If one does not know the weight enumerator of an error control code (which is often the case), an alternative is to can bound the probability of an undetected word error by using the fact that the code can detect all error patters of weight $(d_{min}-1)$ or less. Thus, one can upper bound the probability of undetected word error with the probability that an error pattern with weight d_{min} or greater occurs: [30]

$$P_u(E) \le \sum_{m=d_{min}}^{n} \binom{n}{m} p^m (1-p)^{n-m} = 1 - \left[\sum_{m=0}^{d_{min}-1} \binom{n}{m} p^m (1-p)^{n-m} \right] \tag{4.41}$$

This equation presumes the use of a BSC. One can use the previous results by substituting $P(n,m)$, the probability of m errors occurring in n bits, in for $p^m(1-p)^{n-m}$, which gives the probability that a particular weight m error pattern occurs. This gives:

$$P_u(E) \le 1 - \left[\sum_{m=0}^{d_{min}-1} \binom{n}{m} P(n,m) \right] \tag{4.42}$$

Conversely, the probability of a detected symbol error is bounded by the probability that one or more bit errors occur: [30]

$$P_d(E) \le \sum_{m=1}^{n} \binom{n}{m} p^m (1-p)^{n-m} = 1-(1-p)^n \tag{4.43}$$

Again, one can substitute in for the BSC probability, this time with $P(n,0)$, to get:

$$P_d(E) \le 1 - P(n,0) \tag{4.44}$$

Table 21 gives results for SNR=10dB and Table 22 for SNR=20dB using the length 127 binary BCH codes outlined in Table 20. The “Simulation” and “Analytical” columns list the simulated and analytical probability that each block of 127 bits is received with less than T errors. The goodput, which represents the expected number of correctly received bits per transmitted vector, is then calculated by multiplying this value by K.

Table 21: Performance of length 127 binary BCH codes, FT_c=0.01, SNR=10dB

Binary BCH Code (n,k,t)	Simulation	Analytical	Goodput Simulation	Goodput Analytical
(127,127,0)	0.2496	0.3263	31.6993	41.4365
(127,120,1)	0.4154	0.5095	49.8536	61.1403
(127,113,2)	0.5549	0.6270	62.7073	70.8476
(127,106,3)	0.6694	0.7088	70.9579	75.1307
(127,99,4)	0.7573	0.7694	74.9708	76.1739
(127,92,5)	0.8228	0.8162	75.6980	75.0921
(127,85,6)	0.8708	0.8532	74.0154	72.5186
(127,78,7)	0.9055	0.8827	70.6310	68.8509
(127,71,9)	0.9486	0.9256	67.3518	65.7199
(127,64,10)	0.9621	0.9411	61.5723	60.2282
(127,57,11)	0.9719	0.9535	55.3979	54.3478
(127,50,13)	0.9846	0.9713	49.2285	48.5675
(127,43,14)	0.9886	0.9777	42.5097	42.0395
(127,36,15)	0.9916	0.9827	35.6963	35.3759
(127,29,21)	0.9986	0.9966	28.9595	28.9008
(127,22,23)	0.9992	0.9981	21.9832	21.9580
(127,15,27)	0.9998	0.9994	14.9968	14.9917
(127,8,31)	0.9999	0.9999	7.9995	7.9988

Figure 25 and equation (4.20) showed that the raw BER for a Rayleigh fading channel decreases linearly with increasing SNR. In addition to this difficulty, it appears that the tails on the distribution of errors in a block decay very slowly.

The weight distribution of only a select group of binary BCH codes are known[6] [40]. One can use these formulas to do good work in specific cases, but cannot use them to compare the performance of a wide range of codes. Furthermore, the undetected error probability from (4.42) is not very tight.

One can, however, advance the analysis by considering the case where the wireless link carries data for some higher layer application that utilizes the TCP/IP protocol suite. In this case, a 32-bit CRC check verifies the contents of the datagram (which will have been fragmented across several transmitted vectors) before passing it up the TCP/IP stack. Datagrams that fail the CRC check are dropped.

[6] $T = 2$, $N = 2^m - 1$, $m \geq 3$, m odd; $T = 2$, $N = 2^m - 1$, $m \geq 4$ m even; $T = 3$, $N = 2^m - 1$, $m \geq 5$, m odd; and $T = 3$, $N = 2^m - 1$, $m \geq 6$, m even

Table 22: Performance of length 127 binary BCH codes, FT_c=0.01, SNR=20dB

Binary BCH Code (n,k,t)	Simulation	Analytical	Goodput Simulation	Goodput Analytical
(127,127,0)	0.8039	0.8500	102.1009	107.9552
(127,120,1)	0.9249	0.9271	110.9919	111.2487
(127,113,2)	0.9716	0.9604	109.7910	108.5221
(127,106,3)	0.9884	0.9780	104.7686	103.6687
(127,99,4)	0.9946	0.9878	98.4662	97.7880
(127,92,5)	0.9972	0.9932	91.7436	91.3740
(127,85,6)	0.9985	0.9962	84.8713	84.6795
(127,78,7)	0.9992	0.9979	77.9342	77.8376
(127,71,9)	0.9997	0.9994	70.9794	70.9555
(127,64,10)	0.9998	0.9997	63.9890	63.9781
(127,57,11)	0.9999	0.9998	56.9944	56.9894
(127,50,13)	1.0000	0.9999	49.9981	49.9973
(127,43,14)	1.0000	1.0000	42.9990	42.9988
(127,36,15)	1.0000	1.0000	35.9995	35.9995
(127,29,21)	1.0000	1.0000	29.0000	29.0000
(127,22,23)	1.0000	1.0000	22.0000	22.0000
(127,15,27)	1.0000	1.0000	15.0000	15.0000
(127,8,31)	1.0000	1.0000	8.0000	8.0000

Consider the case of 500 byte packets that are fragmented across independent codewords and transmitted across the channel. The number of codewords required to transmit the 500 bytes varies with the minimum distance of the code: as the minimum distance increases, less "room" for data remains in each codeword due to the increasing number of parity bits. The results appear in Table 23 and are displayed graphically in Figure 51.

If the application can operate acceptably with one corrupted packet per ten transmitted packets, the (127,85,6) BCH code is the highest throughput code that would meet the requirements.

Consider again the length 255 Reed-Solomon code that was initially examined for use across an AWGN channel by examining its performance when used for carrying data across the FT_c=0.01 channel. The GF(256) symbols are converted to binary and transmitted using BPSK as in the previous example. A minimum of eight bits are transmitted between each state transmission, i.e., the channel is used to transmit eight bits before it is checked for a state change. Hence, the symbol error probability is calculated from the bit error

Table 23: Performance of length 127 binary BCH codes, 500 byte packets, FT_c=0.01, SNR=20dB

Binary BCH Code (n,k,t)	Simulation	Analytical	CWs per Datagram	Datagram Simulation	Datagram Simulation
(127,127,0)	0.8039	0.8500	32	0.0009	0.0055
(127,120,1)	0.9249	0.9271	34	0.0704	0.0762
(127,113,2)	0.9716	0.9604	36	0.3545	0.2333
(127,106,3)	0.9884	0.9780	38	0.6415	0.4295
(127,99,4)	0.9946	0.9878	41	0.8012	0.6035
(127,92,5)	0.9972	0.9932	44	0.8844	0.7405
(127,85,6)	0.9985	0.9962	48	0.9299	0.8342
(127,78,7)	0.9992	0.9979	52	0.9570	0.8973
(127,71,9)	0.9997	0.9994	57	0.9836	0.9649
(127,64,10)	0.9998	0.9997	63	0.9892	0.9787
(127,57,11)	0.9999	0.9998	71	0.9931	0.9869
(127,50,13)	1.0000	0.9999	80	0.9970	0.9957
(127,43,14)	1.0000	1.0000	94	0.9979	0.9973
(127,36,15)	1.0000	1.0000	112	0.9984	0.9983
(127,29,21)	1.0000	1.0000	138	0.9999	1.0000
(127,22,23)	1.0000	1.0000	182	1.0000	1.0000
(127,15,27)	1.0000	1.0000	267	1.0000	1.0000
(127,8,31)	1.0000	1.0000	500	1.0000	1.0000

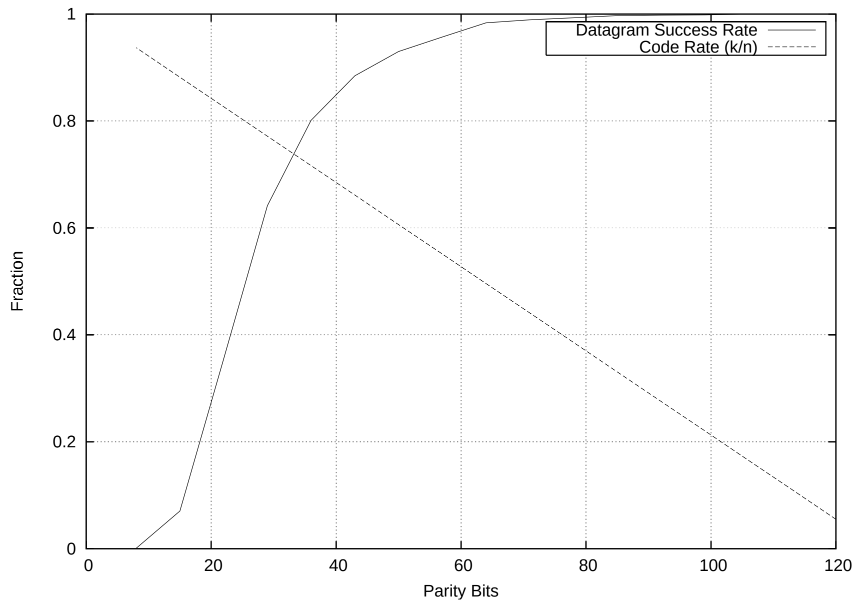

Figure 51: Performance of length 127 binary BCH codes, 500 byte packets, FT_c=0.01, SNR=20dB

probability by taking the probability that all eight bits are received correctly. For example, a bit error probability of 0.05 leads to a symbol error probability of $1-(1-0.05)^8 = 0.3366$. Note that this leads to an interesting consequence: unlike the case of binary codes, it is quite possible to have a symbol error probability of greater than 0.5: a bit error probability of 0.2, for example, leads to a symbol error probability of $1-(1-0.2)^8 = 0.8322$.

Begin by calculating the probability that each block contains less than a given number of errors. These results appear in Figure 52. The bottom curve shows the probabilities at an SNR of 0dB. Each higher curve represents a 3dB improvement in SNR, so that the top curve shows the probabilities at 30dB.

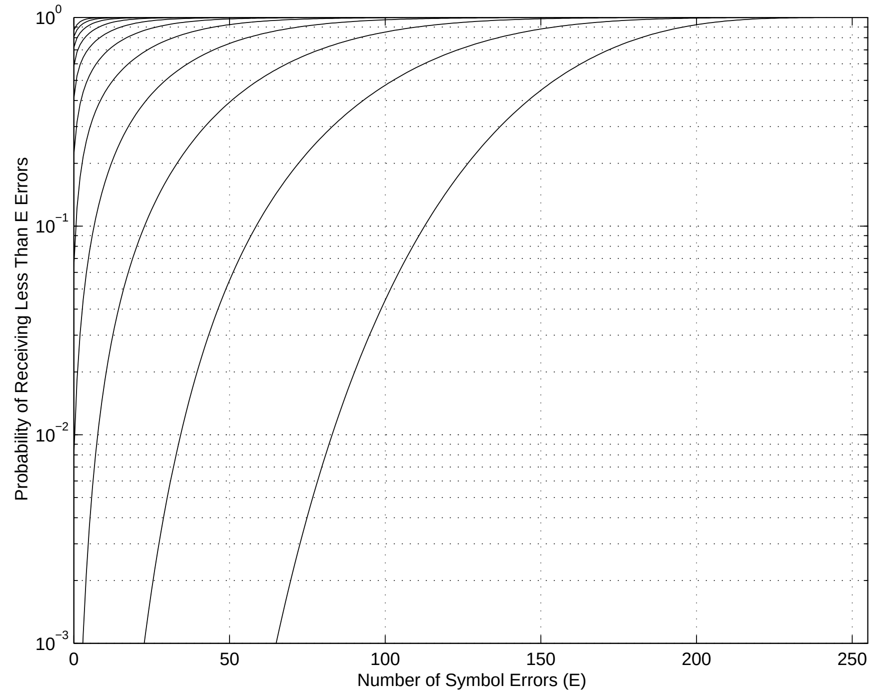

Figure 52: Probability of receiving less than E errors in 255 symbols, FT_c=0.01, SNR=0dB (bottom trace), each step=3dB

Note that at SNR=0dB, the average number of errors in the received vector equals 155! The number of errors quickly trails off to more manageable proportions at higher SNRs; it equals 30 at 9dB and 5 at 15dB.

One can then apply the optimization methodology developed earlier in the chapter. The following example will consider the case for SNR=18dB. The following tables do not show every possible Reed-Solomon code since a Reed-Solomon code exists for every $255 \leq K \leq$

0, $0 \leq T \leq 127$.

Table 24: Performance of length 255 Reed-Solomon codes, FT_c=0.01, SNR=18dB

RS Code (n,k,t)	Code Rate	Vector Success Rate	Vector Goodput	BER Upper Bound
(255,253,1)	0.9922	0.5247	132.7365	4.7535e-01
(255,249,3)	0.9765	0.6446	160.5047	5.9234e-02
(255,243,6)	0.9529	0.7466	181.4142	3.5200e-04
(255,237,9)	0.9294	0.8141	192.9381	5.1233e-07
(255,231,12)	0.9059	0.8632	199.3910	2.8567e-10
(255,225,15)	0.8824	0.8995	202.3977	7.6819e-14
(255,219,18)	0.8588	0.9265	202.9141	1.1473e-17
(255,213,21)	0.8353	0.9465	201.6104	1.0466e-21
(255,207,24)	0.8118	0.9612	198.9780	6.2461e-26
(255,201,27)	0.7882	0.9720	195.3798	2.5679e-30
(255,195,30)	0.7647	0.9799	191.0834	7.5720e-35
(255,189,33)	0.7412	0.9856	186.2852	1.6542e-39
(255,183,36)	0.7176	0.9898	181.1282	2.7496e-44
(255,177,39)	0.6941	0.9927	175.7164	3.5552e-49
(255,171,42)	0.6706	0.9949	170.1245	3.6438e-54
(255,165,45)	0.6471	0.9964	164.4062	3.0086e-59

Unlike the case with the AWGN channel, one finds the code which gives the maximum throughput not by differentiating throughput functions, but by testing possible codes. For this example, the highest-throughput code lies between the (255,237,15) and the (255,225,21) codes. A more detailed look at the codes, as shown in Figure 25, finds that the maximum throughput code is the (255,233,11) code. This code gives a residual BER of less than 2.2933e-16.

When using the (255,231,11) code, one expects that about one out of every 12 received vectors will contain more than 11 errors ($1/(1-0.9184) = 12.2549$). Whether or not this will cause problems will depend on the application. In any case, decoder failures may be traded for throughput as necessary. If the application can tolerate an average of one decoder failure per hundred received vectors, it could use a (255,181,37) code, which has a goodput of 179.3484 bytes per received vector. Or, for one decoder failure per thousand or ten thousand received vectors, it could use a (255,143,56) or (255,107,74) code, which have respective goodputs of 142.8637 and 106.9897 bytes per received vector. These all are

Table 25: Performance of length 255 Reed-Solomon codes, centered around maximum throughput, FT_c=0.01, SNR=18dB

RS Code (n,k,t)	Code Rate	Vector Success Rate	Vector Goodput	BER Upper Bound
(255,237,9))	0.8824	0.8995	202.3977	7.6819e-14
(255,235,10)	0.8745	0.9095	202.8090	4.3275e-15
(255,233,11)	0.8667	0.9184	202.9730	2.2933e-16
(255,231,12)	0.8588	0.9265	202.9141	1.1473e-17
(255,229,13)	0.8510	0.9339	202.6544	5.4346e-19
(255,227,14)	0.8431	0.9405	202.2138	2.4444e-20
(255,225,15)	0.8353	0.9465	201.6104	1.0466e-21

significant reductions in throughput compared to the maximum throughput code, but the goals in each case are different – it makes little sense to compare codes that were chosen to achieve different goals.

The calculations for the delay for an ARQ system will depend on the rate at which packets are transmitted in comparison to the fade rate. If the packets are transmitted at a rate greater than the coherence bandwidth of the channel, then the probabilities for successful transmission must be individually calculated using the matrix probabilities. Hence, one finds the probability that the packet is transmitted once without detected errors

If the packets are transmitted slower than the coherence bandwidth of the channel, then the channel is effectively randomized between transmissions and the reception of the packets may be considered independently.

In this case, the delay for an ARQ system may be calculated using the vector success rate. Assuming perfect feedback, this delay has an exponential distribution, since the transmitter will resend the packet until the receiver returns a positive acknowledgement. The average delay is therefore equal to the mean value of the exponential distribution, which is the inverse of the average success rate. The standard deviation of an exponential distribution is equal to the mean. This technique can be applied to more relistic system models, such as those with unreliable feedback or go-back-N error recovery (which is very useful for analyzing the performance of TCP-based systems) using the techniques outlined in [41] and [42].

The calculation results for FTc=0.01, SNR=18dB, length=255 symbols are tabulated in table 26. The table does not account for a mechanism to detect erroneously decoded packets (such as a CRC check), which would certainly be needed for the higher rate codes. It also includes the case with no error control, which is listed as a (255,255,0) code.

Table 26: ARQ performance using length 255 Reed-Solomon codes, FT_c=0.01, SNR=18dB

RS Code (n,k,t)	Code Rate	Vector Success Rate	Average Delay (Packets)	Good Octets Per Packet
(255,255,0)	1.0000	0.4089	2.4456	104.2695
(255,253,1)	0.9922	0.5247	1.9059	132.7491
(255,249,3)	0.9765	0.6446	1.5513	160.5054
(255,243,6)	0.9529	0.7466	1.3394	181.4238
(255,237,9)	0.9294	0.8141	1.2284	192.9417
(255,231,12)	0.9059	0.8632	1.1585	199.3992
(255,225,15)	0.8824	0.8995	1.1117	202.3875
(255,219,18)	0.8588	0.9265	1.0793	202.9035
(255,213,21)	0.8353	0.9465	1.0565	201.6045
(255,207,24)	0.8118	0.9612	1.0404	198.9684
(255,201,27)	0.7882	0.9720	1.0288	195.3720
(255,195,30)	0.7647	0.9799	1.0205	191.0805
(255,189,33)	0.7412	0.9856	1.0146	186.2784
(255,183,36)	0.7176	0.9898	1.0103	181.1334
(255,177,39)	0.6941	0.9927	1.0074	175.7079
(255,171,42)	0.6706	0.9949	1.0051	170.1279
(255,165,45)	0.6471	0.9964	1.0036	164.4060

4.7 Adaptation

The optimal codes found in the previous chapter remain optimal only so long as the channel conditions do not change. A change in SNR, fade rate, or fading environment may result in another code becoming the optimal. This becomes an especially important issue in Mobile Ad-hoc Wireless Networks (MANETs), where packets may be forwarded multiple times across wireless links. The nodes in a MANET are free to move, which leads to frequent variations in the received signal strength due to propagation losses, intervening terrain features, and interference from other nodes' transmissions.

A high error rate means that packets must be transmitted multiple times before they are received correctly, or that more powerful error correction coding must be used. The use of a fixed error control code will necessarily sacrifice a large measure of performance under certain channel conditions. For example, GSM uses fixed error control codes of different strengths to protect different parts of each voice packet. These codes were chosen to provide good coverage given intended transmitter powers, antenna gains, terrain, and cell sizes.

Wireless Ethernet (802.11b) does not use an error control code. A CRC is used to check the received frame for errors: the frame is dropped if any are found and the receiver does not send an acknowledgment (ACK) packet back to the transmitter. The transmitter then times out and retransmits the packet (typically a maximum of eight times) in the hope that the packet will be received correctly. An error-free acknowledgment is then required from the receiver to complete the exchange. Hence, 802.11b tends to have exhibit a distinct SNR threshold, where throughput drops off very rapidly below the threshold SNR. Errors become very likely and most frames are dropped due to the large number of bits transmitted in each frame. This effect is seen in Figure 4, where the throughput drops off sharply below a threshold SNR (the uncoded case is not shown in this figure, but it would have a high-SNR throughput greater than that for the (240,230) code and begin falling off at a higher SNR). Since beaconing/neighbor discovery/hello packets would be dropped as well, the link would be flagged as unusable to higher-layer applications.

When operating in noisy environments or at longer ranges, 802.11b radios use dynamic rate shifting, which automatically adjusts data rates according to channel conditions. Users would ideally receive the full 11 MBPS data rate, however, if the radios are moved apart or interference is present, the radios will fall back to lower speeds of 5.5, 2, or 1 MBPS. This permits continued operation, but does not take advantage of the coding gain offered by error control codes.

802.11a uses a rate 1/2 convolutional code ($g_0 = 133$ and $g_1 = 171$) which is punctured to give a transmitted data rates of 1/2, 2/3, or 3/4 of the transmitted bit rate.

This threshold effect is common in many modern communications systems. By the time it has left the host computer, data is typically wrapped in at least three checksums and/or CRCs: TCP/UDP, IP, and MAC (some of which only protect the header). An error in any one of these checksums will cause the packet to be dropped.

A fixed error control code pushes the threshold lower, but sacrifices performance above the threshold. This effect can also be seen in Figure 4, where the (240,220) code offers higher throughput below the threshold, but looses out to the higher rate (240,230) code above the threshold.

The use of error control coding would allow for link utilization below this threshold (albeit at a rate lower than the ideal uncoded case) by switching to an error control code that could correct enough errors to pass higher-layer packets consistently. Thus, an 802.11 NIC utilizing adaptive error control would exhibit a decreasing data rate as it moved away from its neighbors, instead of the links suddenly becoming unusable.

An ideal adaptive error control system would offer multiple codes and switch between them as channel conditions changed in an effort to meet the given QoS requirements. However, picking the set of error control codes and the error control code to use at any given time is a difficult problem. Readings of channel SNR and the number of errors in a packet tends to vary from sample to sample due to the random distribution of the number of errors that occur within the packet. Hence, it is important to filter several readings to estimate the channel conditions. Otherwise, the system will attempt to compensate for changes in channel conditions that occur faster than the adaptive error control feedback loop can compensate, such as the case where the adaptive error control system attempts to track fast fading, or chase the "noise" in the signal. Averaging together a large number of readings creates its own set of problems in that it slows the system's reaction time, hampering its ability to react to fast changes in channel SNR, such as when a mobile user passes behind a building.

An adaption algorithm generally requires a BER estimation technique to assist in choosing the appropriate error control code. Many methods have been proposed for determining the BER of a channel. Four of these methods include:

1. Computing the autocorrelation of a Pseudo-random Noise (PN) sequence [43]. A PN sequence is frequently embedded in transmitted packets to provide receiver synchronization and a training sequence for a channel equalizer. Hence, this method

of channel BER estimation does not require the transmission of any additional bits. Upon receiving the packet, the receiver computes the autocorrelation between the received PN sequence and a local copy. This value is then used to update the receiver's ongoing estimate of the channel BER. The accuracy of the estimate depends on the length of the PN sequence (longer sequences lead to more accurate estimates) and the number of sequences received by the receiver.

2. Comparing Viterbi Algorithm metrics. The path metrics computed by the Viterbi algorithm may be compared and used to estimate the channel BER. As the BER increases, the difference between the path metric value of the surviving and losing path entering each state decreases. Similarly, as the BER improves, the metric value of the surviving and losing paths are separated by a greater amount.

3. Counting the symbols corrected by Reed-Solomon or BCH codes [43]. The degree of the error locater polynomial computed during the decoding process gives a reliable estimate of the number of errors present in the received vector. This method obviously fails in the case of a decoder failure: all that is known is that the number of errors exceeds the code's minimum distance. Decoder errors are relatively infrequent events. Therefore, they have little effect on the channel BER estimate, especially as the minimum distance of the code increases. The decoder error and failure probabilities can also be worked into the BER estimation equation. Frequent decoder failures are a clear sign that a more powerful error control code is required given the current channel conditions.

4. Re-encoding convolutionally or block coded data. A quick method for estimating the number of errors in a received vector is to re-encode the decoded data and compare the resulting codeword with the actual received vector. This method is particularly useful for block codes that to not make use of an error locater polynomial (e.g., Golay or Reed-Muller codes) or convolutional codes not decoded with the Viterbi algorithm (e.g., sequential or majority logic decoders). Otherwise, the other techniques can accomplish the same goal with less overhead.

5. Observation of higher-layer effects. A method for estimating channel conditions when the lower-layer processes are not directly observable is to work backward from high-layer effects. For example, a mobile node may transmit beaconing packets every five seconds. A neighboring node could estimate the link quality by observing the fraction of beaconing packets it successfully receives. This mechanism is suitable for comparing multiple links and choosing the most reliable one.

The relationship between channel SNR and the error distribution is not always as simple as the AWGN case given in Equation 4.38. The modeling techniques described in this dissertation may be used to analyze the relationship between channel SNR and error distribution for a range of wireless environments in which the system is expected to operate. In addition, a novel technique that utilizes a pair of moving averages ("dual average") for estimating channel conditions and adapting error control codes was investigated.

Slow changes are fairly easily tracked by a channel SNR estimation process. A successful approach was to estimate the average (μ) and standard deviation (σ) of the number of erroneous symbols in each received packet. The system then used an error control code capable of correcting up $\mu + x \cdot \sigma$ errors in each received block, where x is chosen based on QoS requirements.

The difficulty lies in determining when the channel SNR makes a rapid change. A rapid change is one that takes place within fewer samples (e.g., received packets) than the estimation processes' shortest time constant.

The received SNR and decoded BER may vary considerably between packets due to statistical variations and short term fluctuations in the channel (such as fast fading), and it is therefore difficult to tell whether a packed decoded with significantly more or less errors than expected is indicative of a change in channel quality or merely a statistical fluke. The difficulty lies in recognizing when a rapid but long-term change had occurred (e.g., entering or leaving an urban canyon) and adjusting the error control system to compensate before too many packets are lost.

Looking for trends in the first and second derivative of the received signal strength and BER works well in theory. However, taking derivatives of real world signals tends to be a nightmare because of the measurement noise which is seemingly always present. Computing a derivative often amplifies the noise to the point where it drowns out any useful information. This proved to be the case in the simulations run using this approach despite the use of several different differentiating filters. In each case the noise overwhelmed the sought after SNR information.

Much more success was obtained by using two moving averages to detect sudden SNR changes: a long term average and a short term average. The tests computed two averages of the number of received errors. The long term average was typically calculated over several tens of samples and the short term average over five to ten samples (but in all cases longer than the fade period; attempting to track the fading process is unadvisable). The long term average was used to estimate the channel characteristics (μ and σ) for the purposes of picking a code. The short term average was then compared to long term average in an attempt to recognize rapid changes in channel conditions. A significant rise of the short term average above the long term average tended to indicate a decrease in channel SNR, and vice versa. This was used to trigger an immediate increase or decrease in the minimum distance of the error control code (via a feedback channel) until the two averages were once again more or less equal, indicating a stable channel. At this point, the system could once again rely on the channel estimate from the long-term average.

Tests were conducted using a simulated AWGN channel and Reed-Solomon codes. The code was chosen to maximize throughput. The channel would periodically change SNR by a few dB to simulate channel changes due to mobility (e.g., entering or leaving an urban canyon), and the system would recognize the changes when the short-term SNR estimate exceeded or dropped below the long-term one by a user-defined threshold and switch codes with minimal lost (or overprotected) packets. Other tests involved slow changes in SNR. These were caught by the long term channel estimation process since the short- and long-term estimates did not diverge significantly.

Figures 53 and 54 give examples that show the long term average, short term average, error correcting ability (t) of the Reed-Solomon code, and the number of symbol errors in each received codeword. In figure 53, the SNR improves increases from 3 dB to 5 dB after

250 vectors have been received, while in 54, it increses from 3 dB to 5 dB and then slowly decreases back to 3 dB between symbols 250 and 350. In both cases, the short term average is taken over 10 symbols and the long term average over 100.

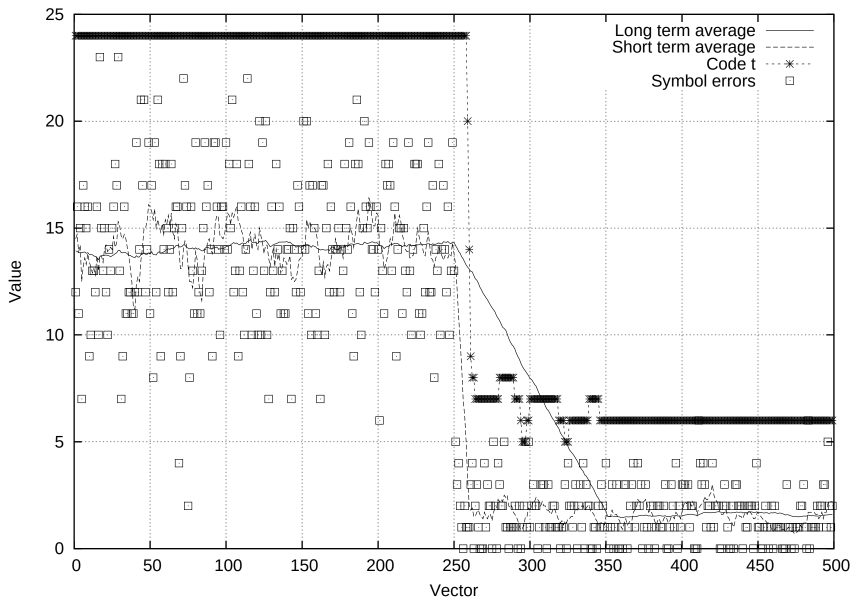

Figure 53: SNR change at 100

The difficulty lies in choosing an appropriate number of packets over which to compute the long and short term averages and picking appropriate thresholds above and below the long term average. These values depend on the the speed and magnitude of the SNR changes. In addition, as the SNR decreases, the number of errors in each received packet tends to become more widely distributed, complicating the process of determining the channel state. In the case of an AWGN channel, the errors describe a binomial distribution around a central mean, but this is not always the case for fading channels. The Markov channel models could prove useful in investigating these distributions and developing heuristics for estimating the SNR since they are excellent tools for calculating the expected distribution of errors across a range of SNRs.

A final briefly-investigated approach involved considering the likelihood of receiving packets with a certain number of errors given the current channel estimate. Reception of several packets with significantly more than $\mu + 2\sigma$ or less than $\mu - 2 * \sigma$ errors is extremely unlikely and generally indicative of a change in those channel characteristics. As was the

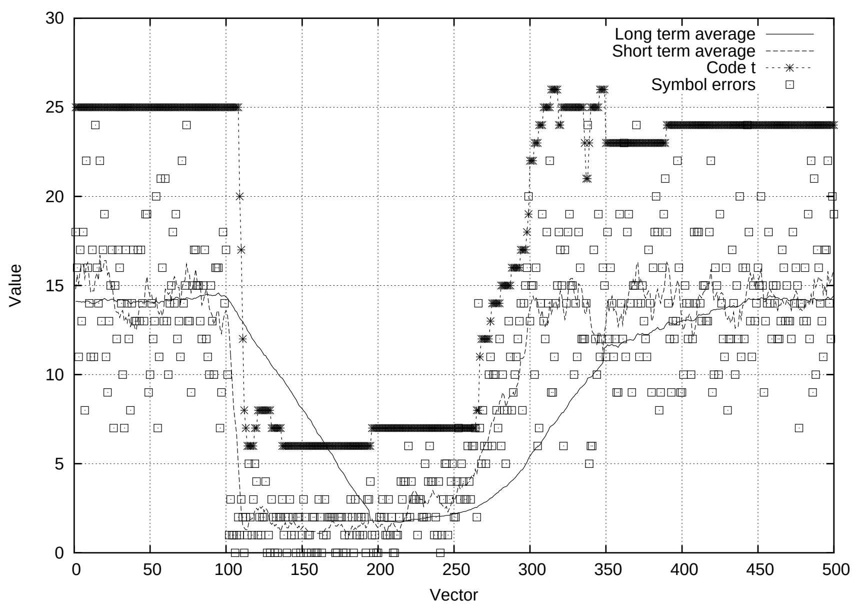

Figure 54: SNR change at 100, gradual increase 250 to 350

case with the dual average method, performance was dependent on matching the thresholds to the channel characteristics.

4.8 Decoding Complexity

This section investigates the processing demands of several error control codes. The computational requirements of decoding an error control code can be an important factor when designing a communication system involving battery powered nodes, particularly if the algorithms will run on general purpose hardware. Battery powered general-purpose hardware, such as handheld computers, and extremely power-conscious devices, such as microsensors, are not well suited to the computational demands of decoding complex error control codes, particularly at high data rates.

However, decoding complexity is rapidly becoming a non-issue in many circumstances due to the availability of inexpensive microchips for decoding error control codes. The time taken by most of these chips to decode a received packet does not depend on the code's minimum distance or constraint length. For example, Advanced Hardware Architecture's Reed-Solomon chips correct codes with minimum distance up to twenty, and its Viterbi decoder chips correct codes with constraint length up to seven, with a delay that is independent of the code.

4.8.1 Convolutional Codes – Viterbi Algorithm

The Viterbi decoder performs a number of different functions during each cycle. These include the addition of branch metrics, the comparison of branch metrics, and the selection of the best branch to each state(the survivor). The following discussion will use m to represent the number of memory elements in the encoder, k the number of encoder inputs, n the number of encoder outputs, N the number of bits in the decoded vector, b_q the number of bits used to quantize the received signal and b_m the number of bits used to store the path metrics.

Each stage of the decoding process, of which there are N/k (not counting trellis edge effects), requires several different calculations. The decoder must add 2^k branch metrics to the partial path metric associated with each of the 2^m nodes. This requires $2^k \cdot 2^m$ additions. The decoder must also choose the path with the lowest metric entering each node, an operation that requires $\left(2^k - 1\right) 2^m$ comparisons at each stage.

The path metrics are simply the inner product product of the received vector and the code words. Each stage therefore requires $2^m \cdot k \cdot n$ b_q-bit multiplications. A clever algorithm could take advantage of the fact that there will be at most 2^n different metrics to calculate at each stage, but such optimizations are beyond the scope of this analysis.

Multiplying two b-bit words requires $7b^2 - 10b$ BOPs, adding them requires $6b - 4$ BOPs, and comparing them requires b BOPs. [44] Thus, decoding a packet requires:

$$\begin{aligned}
\text{comparison BOPs} &: \tfrac{N}{k}\left(2^k - 1\right) 2^m \cdot b_m \\
\text{addition BOPs} &: \tfrac{N}{k} \cdot 2^k \cdot 2^m \left(6b_m + 4\right) \\
\text{multiplication BOPs} &: \tfrac{N}{k} \cdot 2^m \cdot k \cdot n \left(7b_q^2 - 10b_q\right)
\end{aligned}$$

For a total of:

$$\text{BOPs} = \frac{N2^m}{k}\left(\left(2^k - 1\right) \cdot b_m + 2^k \left(6b_m + 4\right) + n\,k\left(7b_q^2 - 10b_q\right)\right) \tag{4.45}$$

One can write the number of BOPs per decoded bit as:

$$\text{BOPs/bit} = \frac{2^m}{k}\left(\left(2^k - 1\right)\cdot b_m + 2^k\left(6b_m + 4\right) + n\,k\left(7b_q^2 - 10b_q\right)\right) \tag{4.46}$$

4.8.2 Convolutional Codes – Sequential Decoding

The authors in [45] showed that the CDF for the number of computations (C) required to decode one information bit for a sequential decoding algorithm has a Pareto distribution:

$$P(C \geq L) \simeq AL^{-\rho} \tag{4.47}$$

where A is a constant that depends on the sequential decoding algorithm and ρ is the Pareto exponent. The Pareto exponent depends on the code rate and channel conditions. It is given by:

$$R = \frac{E_0(\rho)}{\rho} \tag{4.48}$$

where R is the code rate and $E_0(\cdot)$ is the Gallager function, which, for BPSK, equals:

$$E_0(\rho) = -\log_2\left[\frac{1}{2}\sum_{j=0}^{J}\left[P(j|0)^{\frac{1}{1+\rho}} + P(1|j)^{\frac{1}{1+\rho}}\right]^{1+\rho}\right] \tag{4.49}$$

where $P(j|k)$ is the probability of receiving j when k was sent. In the event of a binary symmetric channel (BSC), (J in (4.49) equals two, e.g., an AWGN channel and hard decision decoding) one can further reduce (4.49) to:

$$E_0(\rho) = -\log_2\left[\epsilon^{\frac{1}{1+\rho}} + (1-\epsilon)^{\frac{1}{1+\rho}}\right]^{1+\rho} \tag{4.50}$$

where ϵ is the crossover probability (probability of error). In the case of soft decision decoding (i.e., $J \to \infty$), this gives:

$$E_0(\rho) = -\log_2\left[\frac{1}{2}\int\limits_{j=-\infty}^{+\infty}\left[P(j|0)^{\frac{1}{1+\rho}} + P(1|j)^{\frac{1}{1+\rho}}\right]^{1+\rho} dj\right] \tag{4.51}$$

4.8.3 Reed Solomon Codes

The decoding complexity of Reed-Solomon codes depends on a number of factors, including code length, error correcting ability and the decoding algorithm. Several techniques exist for solving the key equation; these include the Peterson-Gorenstein-Zierler (PGZ) algorithm [46], the Berlekamp-Massey Algorithm (BMA) [47] [48], Euclid's algorithm [30], and the Galois Field Fourier Transform approach [30]. The following evaluation, which is excerpted from [44], assumes the use of the BMA to solve the key equation, the Chien search [49] to locate the errors and the Forney algorithm [50] to compute the magnitude of the errors.

$$\begin{aligned} \text{GF mults} &= \tfrac{t}{2}\left(19t + 7n - 3\right) \\ \text{GF adds} &= \tfrac{t}{2}\left(19t + 7n - 5\right) \end{aligned} \tag{4.52}$$

The authors in [44] roughly convert GF operations to BOPs by assuming that a GF addition costs m BOPs and a GF multiplication $2m\,(2m-1)$ BOPs. This gives a decoding cost of:

$$\text{BOPs} = \frac{tm}{2}\left(76tm + 28nm - 12m - 19t - 7n + 1\right) \tag{4.53}$$

Each successfully decoded vector results in mk decoded bits. Hence, one can write the decoding cost in BOPs per bit as:

$$\text{BOPs/bit} = \frac{t}{2k}\left(76tm + 28nm - 12m - 19t - 7n + 1\right) \tag{4.54}$$

which makes a good figure of merit for comparing Reed-Solomon codes of disparate lengths.

These equations can be used to examine the computational requirements of the the Reed-Solomon codes investigated in Section 4.6. The processing requirements for the Reed-Solomon codes listed in Table 24 appears in Table 27.

Table 27: Binary operations to decode Reed-Solomon packets, FTc=0.01, SNR=18dB

RS Code		Binary Operations	Binary Operations
(N,K,T)	Code Rate	to Decode Packet	Per Decoded Bit
(255,253,1)	0.9922	223316	111
(255,249,3)	0.9765	684084	344
(255,243,6)	0.9529	1410576	726
(255,237,9)	0.9294	2179476	1150
(255,231,12)	0.9059	2990784	1619
(255,225,15)	0.8824	3844500	2136
(255,219,18)	0.8588	4740624	2706
(255,213,21)	0.8353	5679156	3333
(255,207,24)	0.8118	6660096	4022
(255,201,27)	0.7882	7683444	4779
(255,195,30)	0.7647	8749200	5609
(255,189,33)	0.7412	9857364	6520
(255,183,36)	0.7176	11007936	7520
(255,177,39)	0.6941	12200916	8617
(255,171,42)	0.6706	13436304	9822
(255,165,45)	0.6471	14714100	11148

CHAPTER V

CONCLUSIONS AND FUTURE RESEARCH

5.1 Conclusions

This dissertation addresses channel modeling and techniques for choosing codes to meet QoS requirements in an adaptive error control system. Chapter 2 gives an overview of related work and 3 presents relevant material from prior investigations into adaptive error control and channel modeling. Chapter 4 presents the research findings that form the backbone of the dissertation, including results from the calculations and simulations undertaken in the course of this dissertation. In particular, it includes an analysis of different quantization schemes, Markov models, as well as a discussion of the complexity and performance of error control codes from the viewpoint of QoS guarantees.

Markov modeling of wireless channels has recently received renewed attention in the academic literature. These papers tend to use Markov models to model the block error probability of the channel. This dissertation had taken a different approach, and instead used Markov models to investigate the distribution of errors within a received block. A general model is developed based on a fade rate and quantization scheme. This general model involves fitting the state transition and state holding parameters to a simulated or record fading waveform. The model may then be utilized at an arbitrary SNR to calculate error distributions in the received packets. This approach has the advantage over simulations and semi-analytic methods in that all of the calculations with the models are performed analytically and do not rely on Monte Carlo techniques. However, great care must be taken in fitting the models to sample waveforms, since errors in this state will affect all further calculations performed using the models. The chapter also includes a comparison of different modeling approaches, including Markov models, hidden Markov models, and various quantization schemes. The advantages of logarithmic quantization regions over MMSE quantization regions is shown through a number of examples.

These models were then utilized to calculate error rates and goodputs an FEC system. While Reed-Solomon codes are used as examples, the results can be applied to other error control codes as well. The discussion centered around choosing an error control code to meet error or throughput requirements. ARQ systems were then briefly examined in terms of the delay distribution. Using lower rate codes to protect the data resulted in both lower average delay and a smaller standard deviation for the delay.

The section on adaptive error control suggested the use of the channel models for analyzing error control codes and picking a range of codes that operate effectively across the expected range of error conditions. It then discussed some of the trade-offs inherent in an adaptive error control system. An example system for tracking both fast and slow variations in channel SNR was presented.

Appendix B presented an short tutorial on the matrix calculations used throughout this dissertation and, in particular, discussed the matrix Fourier Transforms that make the calculations for error distributions feasible. Appendix C gave a detailed explanation of the curve fitting techniques involved in calculating the matrices that comprise each of the channel models.

5.2 Future Research

The natural progression from this dissertation is to use the tools developed herein to assist in the design of adaptive error control systems. In particular, the tools can be used to examine the performance of a wide variety of codes across channels which the system may be expected to operate.

Section 4.6 investigated the relationship between channel SNR and the error distribution for an AWGN channel. This relationship is generally not as simple for fading channels, and the equations rarely exist in closed form. The modeling techniques described in this dissertation may serve as a jumping-off point for investigating the nature of these relationships.

This dissertation focussed on Rayleigh fading, which is a good approximation for the behavior of radio frequency signals in macrocellular applications and high-clutter environments. Given the ever-increasing interest in microcells, wireless LANs, and wireless last-mile services, along with the applicability of adaptive error control to these environments, a thorough investigation of Rician, Nakagami, and other fading processes would prove useful. The proposed adaptation system, which uses both a long term and a short term average to track channel SNR changes, also deserves further investigation.

APPENDIX A

WEIGHT AND WEIGHT DISTRIBUTION

The weight of a codeword is equal to the number of nonzero symbols in the codeword. Hence, the codeword [1110001] from a $(7,3)$ systematic binary linear code has weight four.

The weight distribution of a code gives the number of codewords of each weight. It is frequently written as a polynomial in one variable where the exponents specify the weight of the codeword and the coefficients the number of codewords:

$$W(x) = \sum_{i=0}^{n} A_i x^i$$

In this expression, i is the weight and A_i the number of weight i codewords.

The weights of the codewords in the codespace of the BCH (15,7) code are distributed as follows:

Weight	0	5	6	7	8	9	10	15
Codewords	1	18	30	15	15	30	18	1

which gives the following polynomial:

$$x^{15} + 18x^{10} + 30x^9 + 15x^8 + 15x^7 + 30x^6 + 18x^5 + 1$$

The complete weight enumerator provides even more detail in that it distinguishes between the weight of the information and parity sequences. The complete weight enumerator can be written in polynomial form as:

$$W(x,y) = \sum_{i=0}^{n} \sum_{j=0}^{n} A_{i,j} x^i y^j$$

where i is the weight of the information sequence, j is the weight of the parity sequence and $A_{i,j}$ is the number of codewords having this combination of weights.

APPENDIX B

MATRIX PROBABILITIES

B.1 Preliminaries

When faced with a simple channel and system model, a engineer can often compute the performance through direct calculation. Such is the case with, for example, a system using uncoded quaternary phase shift keying (QPSK) modulation across additive white Gaussian noise (AWGN) channel. The derivation of the error rate for this system is within the reach of any engineering student who has taken an introductory telecommunications class.

As the channel and system become more complex, the calculations become more involved and often involve expressions that lack closed form solutions or are simply unknown. Even something as seemingly simple as calculating the residual error rate for a Reed-Solomon code across an AWGN channel cannot be solved exactly. This is true because the complete binary weight enumerator is not known for the majority of Reed-Solomon codes. Hence, an exact numerical value is unobtainable. In other cases, the mathematical expressions describing the system become too complex solve exactly, such as is the case with a system that communicates over a fading multipath channel corrupted by multiple co- and adjacent-channel interferers.

Upon reaching such an impasse, engineers must make simplifying assumptions, resort to bounding operations, simulate the system on a computer or some combination of these three.

Engineers have taken a wide variety of approaches to analyzing communications systems using computer simulation techniques. These range from detailed waveform level simulations that examine actual receiver structure (e.g., GSMsim and [51]) to system level simulations that monitor the traffic between nodes [52]. Such Monte Carlo simulations can provide a high level level of accuracy (measurable with confidence intervals), but often at a prohibitive computational burden [53].

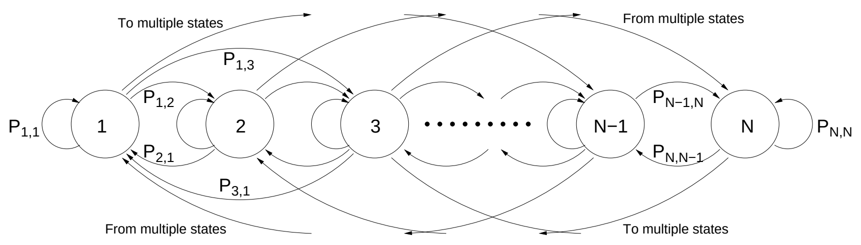

Figure 55: N-state channel

Hidden Markov models (HMM) allow us to overcome many of the problems inherent in regular analytical calculations or simulation. By fitting HMMs to pieces in the communication chain, one can combine the HMMs and arrive at an expression that can be evaluated analytically. If the models are good, one can be sure of the results, and, while the matrices can grow fairly large, the execution time on a computer remains far less than direct simulation.

The simplest example of a HMM for a communications system is the Gilbert channel. The Gilbert channel is simple enough that it can be investigated using some basic tools from linear algebra. While more sophisticated models may also be investigated using linear algebra, the complexity of the algebraic expressions is generally prohibitive, and the expressions are best left in matrix form and evaluated on a computer. This investigation begins by introducing some important concepts in channel modeling with matrices and then examining the Gilbert channel.

B.2 Hidden Markov Model Development

Consider a finite channel with N states $S = \{s_1, s_2, \ldots, s_N\}$. This channel is shown in Figure 55 and is used by Shannon in his derivation of channel capacity [3]. At each time interval t, the channel begins in state $s[t-1] \in S$, accepts some input symbol $a[t] \in A$, transitions to state $s[t]$, and outputs $b[t] \in B$. This occurs with probability $P(s[t], b[t]|s[t-1], a[t])$.

This probability easily expands to cover sequences of symbols. The probability of receiving vector $\mathbf{b} = [b[1], b[2], \ldots, b[T]]$ and ending in state $s[T]$ given transmitted vector $\mathbf{a} = [a[1], a[2], \ldots, a[T]]$ and starting state $s[0]$ is:

$$P(s[n], \mathbf{b}|s[0], \mathbf{a}) = \sum_{s[1]} \cdots \sum_{s[n-1]} \prod_{t=1}^{T} P(s[t], b[t]|s[t-1], a[t]) \tag{B.1}$$

One can eliminate the multiple summations, thereby simplifying this expression considerably, by folding all of the possible state transitions into matrices. This is done using an $N \times N$ matrix:

$$\mathbf{P}(b[t]|a[t]) \tag{B.2}$$

where $P_{i,j}$ is $P(s[t] = j, b[t]|s[t-1] = i, a[t])$. Instead of summing over all possible state transitions, multiply matrices together and let the matrix multiplication take care of the state transitions. The manner in which this occurs is best illustrated by an example:

To save space, let $\mathbf{A}$ represent $\mathbf{P}(b[1]|a[1])$ (the matrix for $t = 1$) and $\mathbf{B}$ represent $\mathbf{P}(b[2]|a[2])$ (the matrix for $t = 2$). Hence, $a_{i,j}$ equals $P(s[1] = j, b[1]|s[0] = i, a[1])$ and $b_{i,j}$ equals $P(s[2] = j, b[2]|s[1] = i, a[2])$. The entries for each matrix appear in the usual fashion:

$$\mathbf{A} = \begin{bmatrix} a_{1,1} & a_{1,2} & \ldots & a_{1,N} \\ a_{2,1} & a_{1,2} & \ldots & a_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N,1} & a_{n,2} & \ldots & a_{N,N} \end{bmatrix} \quad \mathbf{B} = \begin{bmatrix} b_{1,1} & b_{1,2} & \ldots & b_{1,N} \\ b_{2,1} & b_{1,2} & \ldots & b_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ b_{N,1} & b_{n,2} & \ldots & b_{N,N} \end{bmatrix}$$

When multiplied together, $\mathbf{A}$ and $\mathbf{B}$ give:

$$\begin{bmatrix} a_{1,1}b_{1,1} + a_{1,2}b_{2,1} + \cdots + a_{1,N}b_{N,1} & \ldots & a_{1,1}b_{1,N} + a_{1,2}b_{2,N} + \cdots + a_{1,N}b_{N,N} \\ \vdots & \ddots & \vdots \\ a_{N,1}b_{1,1} + a_{N,2}b_{2,1} + \cdots + a_{N,N}b_{N,1} & \ldots & a_{N,1}b_{1,N} + a_{N,2}b_{2,N} + \cdots + a_{N,N}b_{N,N} \end{bmatrix}$$

Notice the combination of terms in each position. In the upper left position, one can add together the probabilities for the path that stays in state one the entire time ($1 \to 1 \to 1$), the path that jumps to state two and then back to state one ($1 \to 2 \to 1$) and so on up to the path that jumps to state N and back to state one ($1 \to N \to 1$). All of the terms in the $(1, 1)$ position represent transitions that begin and end in state one. Similarly, all of the terms in the (i, j) position represent transitions that start in state i and end in state j.

Multiplying together N such matrices gives a similar result; the term in the (i, j) position gives the probability of ending in state j and receiving $\mathbf{b}$ given that the the channel started in state i and that $\mathbf{a}$ was transmitted, i.e., $P(s[T] = j, \mathbf{b}|s[0] = i, \mathbf{a})$.

The probability calculation now looks like:

$$P(\mathbf{b}|\mathbf{a}) = \prod_{t=1}^{T} \mathbf{P}(b[t]|a[t]) \tag{B.3}$$

This gives us, in matrix form, the probability of receiving $\mathbf{b}$ given that $\mathbf{a}$ was transmitted. To extract the desired scalar probability from this matrix probability, one needs to multiply the probability matrix by vectors that will zero out all other entries of the matrix and leave us with the desired scalar probability. This is accomplished by defining unit vectors $\mathbf{s0}$ and $\mathbf{sN}$ where:

$$s0_n = \begin{cases} 1, \ n = i \\ 0, \ n \neq i \end{cases}$$

$$sN_n = \begin{cases} 1, \ n = j \\ 0, \ n \neq j \end{cases}$$

and premultiplying (B.3) by $\mathbf{s0}$ and postmultiplying by $\mathbf{sN}$ to give:

$$P(s[T], \mathbf{b}|s[0], \mathbf{a}) = \mathbf{s0}\left(\prod_{t=1}^{T} \mathbf{P}(b[t]|a[t])\right)\mathbf{sN} \tag{B.4}$$

which gives the same result as (B.1) – the probability of starting from state i and ending in state j while receiving $\mathbf{b}$ given that $\mathbf{a}$ was sent – except that the sum occurs over all possible paths between i and j].

The input and output values are the important items in the the performance analysis of a communications system, not the beginning and ending states of the channel. This requires the calculation of $P(\mathbf{b}|\mathbf{a})$, not $P(s[T] = j, \mathbf{b}|s[0] = i, \mathbf{a})$.

This is accomplished by summing over the probability of the channel ending in all possible states to remove the ending state of the channel from the probability. Replacing postmultiplication column vector $\mathbf{sN}$ with the all ones vector ($\mathbf{1}$) accomplishes this, as postmultiplication by $\mathbf{1}$ is effectively summation: it adds together the probabilities of ending in states 1 through N. Entry j in row vector $\left(\prod_{t=1}^{T} \mathbf{P}(b[t]|a[t])\right)\mathbf{1}$ contains $P(\mathbf{b}|\mathbf{a}, s[0])$.

Removing the condition on the starting state is done similarly. Premultiplication by $\mathbf{1}$ would imply that the channel starts from every possible state with probability one, which is clearly impossible. However, $\sum_{j=1}^{N} P(s[0] = j) = 1$, so one should premultiply with some row vector $\boldsymbol{\pi}$ where $\sum_{j=1}^{N} \pi_j = 1$. This assumes a homogeneous channel.

$\boldsymbol{\pi}$ is often defined as the state probability distribution where π_j is the probability of finding the channel in state j at any given time. If state transition probabilities are time invariant, $\boldsymbol{\pi}$ is the solution to $\boldsymbol{\pi}\mathbf{P}(b|a) = \boldsymbol{\pi} \ni \boldsymbol{\pi}\mathbf{1} = 1$.

Thus,

$$P(\mathbf{b}|\mathbf{a}) = \boldsymbol{\pi}\left(\prod_{t=1}^{T} \mathbf{P}(b[t]|a[t])\right)\mathbf{1} \tag{B.5}$$

Two simplifying assumptions that ease the mathematical development considerably at the price of a small loss of generality are made before moving on to the next phase of the analysis:

- The channel's output is not affected by the input, i.e.,
  $P(s[t], b[t]|s[t-1], a[t]) = P(s[t], b[t]|s[t-1])$
- The transition probabilities are independent of time (the channel is homogeneous), which has already been assumed), i.e.,
  $P(s[t], b[t]|s[t-1]) = P(s[t+k], b[t+k]|s[t+k-1]) \; \forall \, k \in \mathbb{Z}$

This channel becomes extremely if useful if one thinks of the channel outputs as indications of whether the symbol was received successfully or unsuccessfully instead of representing the actual value of the symbol. This turns the model into a powerful tool for evaluating the performance of a communications system in that the channel's output represents an error sequence. This switch is made by examining the sequence of reception errors ($\mathbf{e}$) instead of the sequence of channel outputs ($\mathbf{b}$). Since the channel is time invariant and its output is not affected by the input, probability matrix (B.2) reduces to $\mathbf{P}(e)$. The probability of an error sequence $\mathbf{e}$ is then calculated similarly to the expression in (B.5):

$$P(\mathbf{e}) = \boldsymbol{\pi}\left(\prod_{t=1}^{T}\mathbf{P}(e[t])\right)\mathbf{1} \tag{B.6}$$

where $\mathbf{e}$ is a length T vector of error events. For example:

$$P([010100]) = \boldsymbol{\pi}\mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(0)\mathbf{P}(0)\mathbf{1}$$

The entries of $\mathbf{e}$ are necessarily 1 or 0 since one either has an error or one does not.[1] This gives a very simple expression for the channel BER, which is simply the probility of receiving an error:

$$BER = \boldsymbol{\pi}\mathbf{P}(1)\mathbf{1} \tag{B.7}$$

The next step is to develop matrices $P(1)$ (the matrix error probability) and $P(0)$ (the matrix success probability), which are used to compute channel error and success events. The begin with the state transition matrix $\mathbf{P}$ where $P_{i,j} = P(s[t] = j|s[t-1] = i)$:

$$\mathbf{P} = \begin{bmatrix} P(s[t]=1|s[t-1]=1) & \dots & P(s[t]=N|s[t-1]=1) \\ P(s[t]=1|s[t-1]=2) & \dots & P(s[t]=N|s[t-1]=2) \\ \vdots & \ddots & \vdots \\ P(s[t]=1|s[t-1]=N) & \dots & P(s[t]=N|s[t-1]=N) \end{bmatrix}$$

Each state has a probability of error associated with it. The manner in which this error probability is calculated is determined by the aspects of the channel under consideration represented by the states. The states could represent fading levels in a mobile microwave channel, synchronization in a fiber optical link, atmospheric interference in a point-to-point laser link, or any other processes that that can be modeled as a Markovian process. In the case of fading levels, the error probability is determined by the average error probability at each level.

[1]This does not preclude the use of soft decision decoding, as will be shown in later sections.

The error probability is associated with the state entered by the channel. Thus, one computes $P(1)$ by multiplying column j of P by the error probability for state s_j. Similarly, one computes $P(0)$ by multiplying column j of P by the success probability for state s_j. Once again, these computations can be performed using matrix multiplication, in this case with diagonal matrices $\mathbf{F}(1)$ for error events and $\mathbf{F}(0)$ for successful receptions. $F(1)_{i,i}$ is the error probability for state s_i:

$$\mathbf{F}(1) = \begin{bmatrix} P(error|s[t]=1) & 0 & \dots & 0 \\ 0 & P(error|s[t]=2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & P(error|s[t]=N) \end{bmatrix}$$

and $\mathbf{F}(0) = \mathbf{I} - \mathbf{F}(1)$.

Multiplying $\mathbf{P}$ and $\mathbf{F}(1)$ gives $\mathbf{P}(1)$, the matrix probability for an error event. The resulting matrix ($\mathbf{PF}(1) = \mathbf{P}(1)$) has entries:

$$P(1)_{i,j} = P(s[t]=j|s[t-1]=i)P(error|s[t]=j)$$

and the successful reception matrix $\mathbf{P}(0)$ ($\mathbf{PF}(0) = \mathbf{P}(0)$) has entries:

$$\begin{aligned} P(1)_{i,j} &= P(s[t]=j|s[t-1]=i)(1-P(error|s[t]=j)) \\ &= P(s[t]=j|s[t-1]=i)P(success|s[t]=j) \end{aligned}$$

(B.6) can be used to compute the probability of any error sequence. For example, for $\mathbf{e} = [0110]$, $P(\mathbf{e}) = \boldsymbol{\pi}\mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(1)\mathbf{P}(0)\mathbf{1}$. However, a discussion of the simplest HMM error source is in order before taking a look at matrix probabilities in greater detail.

B.3 The Gilbert Channel

The Gilbert channel was developed by Gilbert [17] to model a telecommunications channel. This channel model may be in one of two states: a good state (G) and a bad state (B). The channel transitions from the good state to the bad state with probability P and remains in the good state with probability Q. Conversely, it transitions from the bad state to the good state with probability p and remains in the bad state with probability q.

The channel passes the data without error when in the good state. In the bad state, the error probability is $1-h$[2]. These error probabilities are independent of the transmitted data. Hence, one can focus on the error events as discussed previously without concern for the actual transmitted data.

The transition probabilities between the good and bad states are illustrated graphically in Figure 56 and as a Markov Model in (B.8). The successful and unsuccessful transmission matrices are shown in (B.9).

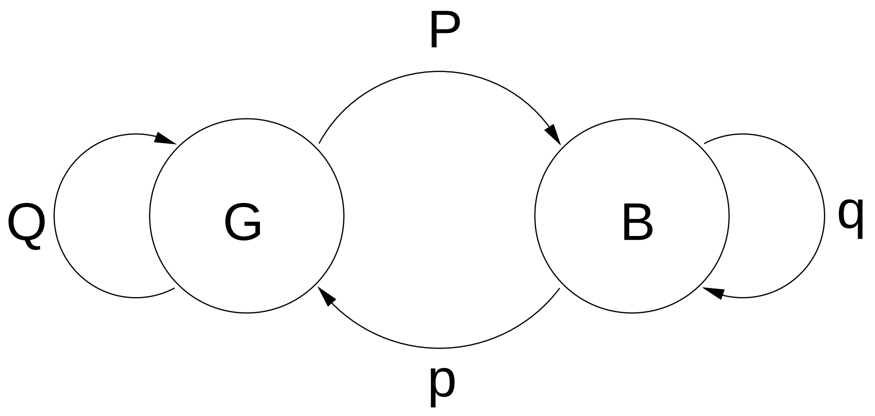

Figure 56: Gilbert channel

$$\mathbf{P} = \begin{bmatrix} Q & P \\ p & q \end{bmatrix} \tag{B.8}$$

$$\mathbf{F}(0) = \begin{bmatrix} 1 & 0 \\ 0 & h \end{bmatrix} \quad \mathbf{F}(1) = \begin{bmatrix} 0 & 0 \\ 0 & 1-h \end{bmatrix} \tag{B.9}$$

One can then obtain $\mathbf{P}(0)$ and $\mathbf{P}(1)$ as shown below:

$$\mathbf{P}(0) = \mathbf{PF}(0) = \begin{bmatrix} Q & Ph \\ p & qh \end{bmatrix} \quad \mathbf{P}(1) = \mathbf{PF}(1) = \begin{bmatrix} Q & P(1-h) \\ p & q(1-h) \end{bmatrix}$$

[2]The error probability is 1 in some discussions of the Gilbert Channel.

The stationary state distribution $\boldsymbol{\pi}$ solves $\boldsymbol{\pi}\mathbf{P} = \boldsymbol{\pi}$ and, for the two state case, expands to:

$$\begin{aligned} \pi_1 Q + \pi_2 p &= \pi_1 \\ \pi_1 Q + \pi_2 p &= \pi_1 \\ \pi_1 + \pi_2 &= 1 \end{aligned}$$

Which one can solve to get:

$$\begin{aligned} \pi_1 &= \frac{p}{P+p} \\ \pi_2 &= \frac{P}{P+p} \end{aligned}$$

Hence,

$$\boldsymbol{\pi} = \begin{bmatrix} \frac{p}{P+p} & \frac{P}{P+p} \end{bmatrix}$$

The instantaneous bit error is calculated in a straightforward fashion:

$$\begin{aligned} P(\text{bit error}) &= \boldsymbol{\pi}\mathbf{P}(1)\mathbf{1} \\ &= \begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix} \begin{bmatrix} 0 & \mathbf{P}(1-h) \\ 0 & q(1-h) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} \frac{p}{P+p} & \frac{P}{P+p} \end{bmatrix} \begin{bmatrix} 0 & \mathbf{P}(1-h) \\ 0 & q(1-h) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\ &= \frac{p}{P+p} P(1-h) + \frac{P}{P+p} q(1-h) \\ &= \frac{P(1-h)(p+q)}{P+p} \\ &= \frac{P(1-h)}{P+p} \end{aligned}$$

These matrices allow one to calculate the probability of various error patterns. For example, the probability of the (10) error pattern may be calculated as:

$$
\begin{aligned}
P(10) &= \boldsymbol{\pi}\mathbf{P}(1)\mathbf{P}(0)\mathbf{1} \\
&= \begin{bmatrix} \pi_1 & \pi_2 \end{bmatrix} \begin{bmatrix} 0 & P(1-h) \\ 0 & q(1-h) \end{bmatrix} \begin{bmatrix} Q & Ph \\ P & qh \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \begin{bmatrix} \frac{p}{P+p} & \frac{P}{P+p} \end{bmatrix} \begin{bmatrix} pP(1-h) & qhP(1-h) \\ qp(1-h) & q^2h(1-h) \end{bmatrix} \begin{bmatrix} 1 \\ 1 \end{bmatrix} \\
&= \frac{1}{P+p}(p^2P(1-h) + qpP(1-h) + pqhP(1-h) + q^2hP(1-h)) \\
&= \frac{P(1-h)}{P+p}(p^2 + qp + pqh + q^h) \\
&= \frac{P(1-h)(p+q)(p+qh)}{P+p} \\
&= \frac{P(1-h)(p+qh)}{P+p}
\end{aligned}
$$

One can find the probability of receiving l error free symbols in a row (i.e., (10^n1)) by calculating:

$$
P(n) = \frac{\boldsymbol{\pi}\mathbf{P}(1)\mathbf{P}^n(0)\mathbf{P}(1)\mathbf{1}}{\boldsymbol{\pi}\mathbf{P}(1)\mathbf{1}}
$$

However, this can be time consuming to calculate for large values of n.

Matrix multiplication is a computationally intensive process. One can save a great deal of time by spectrally decomposing $\mathbf{P}(0)$:

$$
\mathbf{P}(0) = \mathbf{S}\boldsymbol{\Lambda}\mathbf{S}^{-1}
$$

Where $\boldsymbol{\Lambda}$ is a diagonal matrix of eigenvalues and $\mathbf{S}$ is the associated right eigenvector matrix. The eigenvalues solve:

$$
|\lambda\mathbf{I} - \mathbf{P}(0)| = 0
$$

For the 2×2 case one can then write:

$$\begin{vmatrix} \lambda - Q & -Ph \\ -p & \lambda - qh \end{vmatrix} = (\lambda - Q)(\lambda - qh) - pPh$$

$$= \lambda^2 - qh\lambda - Q\lambda + qQh - pPh$$

$$= \lambda^2 - (Q + qh)\lambda + h(Q - p)$$

Hence,

$$\lambda_{1,2} = \frac{Q + qh}{2} \pm \sqrt{\left(\frac{Q + qh}{2}\right)^2 - h(Q - p)}$$

Knowing the eigenvalues, one can then calculate the corresponding right eigenvectors, $\mathbf{x}_1$ and $\mathbf{x}_2$, as the nonzero solutions of

$$\mathbf{P}(0)\mathbf{x} = \lambda\mathbf{x}$$

for $\lambda = \lambda_1$ and λ_2. Substituting in for the 2×2 case gives:

$$[rl] \begin{bmatrix} Q & Ph \\ p & qh \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \lambda \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

$$Qx_1 + Phx_2 - \lambda x_1 = 0 \quad \rightarrow \quad (\lambda - Q)x_1 = Phx_2$$

$$px_1 + qhx_2 - \lambda x_2 = 0 \quad \rightarrow \quad px_1 = (\lambda - qh)x_2$$

Either equation will give a valid pair of eigenvalues. For purposes of this example, choose:

$$\mathbf{x}_1 = \begin{bmatrix} Ph \\ \lambda_1 - Q \end{bmatrix} \quad \mathbf{x}_2 = \begin{bmatrix} Ph \\ \lambda_2 - Q \end{bmatrix}$$

Which gives:

$$\mathbf{S} = \begin{bmatrix} Ph & Ph \\ \lambda_1 - Q & \lambda_2 - Q \end{bmatrix} \quad \mathbf{S}^{-1} = \frac{1}{Ph(\lambda_2 - \lambda_1)} \begin{bmatrix} \lambda_2 - Q & -Ph \\ Q - \lambda_1 & Ph \end{bmatrix}$$

Thus,

$$\begin{aligned}
[rl]\mathbf{S}\mathbf{\Lambda}\mathbf{S}^{-1} &= \frac{1}{|\mathbf{S}|} \begin{bmatrix} Ph & Ph \\ \lambda_1 - Q & \lambda_2 - Q \end{bmatrix} \begin{bmatrix} \lambda_1^n & 0 \\ 0 & \lambda_2^n \end{bmatrix} \begin{bmatrix} \lambda_2 - Q & -Ph \\ -(\lambda_1 - Q) & Ph \end{bmatrix} \\
&= \frac{1}{|\mathbf{S}|} \begin{bmatrix} Ph(\lambda_1^n(\lambda_2 - Q) - \lambda_2^n(\lambda_1 - Q)) & P^2h^2(\lambda_2^n - \lambda_1^n) \\ (\lambda_1^n - \lambda_2^n)(\lambda_1 - Q)(\lambda_2 - Q) & Ph(\lambda_2^n(\lambda_2 - Q) - \lambda_1^n(\lambda_1 - Q)) \end{bmatrix}
\end{aligned}$$

where $|\mathbf{S}| = Ph(\lambda_2 - \lambda_1)$

One can then work through copious quantities of algebra to find:

$$\begin{aligned}
P(n) &= \frac{1-h}{\lambda_1 - \lambda_2}[\lambda_1^n(p - Q + q\lambda_1) - \lambda_2^n(p - Q + q\lambda_2)] \\
&= a(1 - \lambda_1)\lambda_1^n + (1 - a)(1 - \lambda_2)\lambda_2^n
\end{aligned}$$

Where

$$a = \frac{(1-h)(p - Q + q\lambda_1)}{(1 - \lambda_1)(\lambda_1 - \lambda_2)}$$

B.4 Event Probabilities

In most performance calculations, one is interested in the number of errors that occur in a given block of bits and not in the actual location of each error within the block. This brings us to the notion of an error event, where an error event includes a number of individual error patterns.

Consider an event E characterized by the erroneous reception of one out of three symbols. This can occur in three possible ways: (100), (010) or (001). Therefore, $E = \{(100) \cap (010) \cap (001)\}$. The matrix probability that E occurs is therefore the sum of the matrix probabilities for each of these three events:

$$\mathbf{P}(E) = \mathbf{P}(1)\mathbf{P}^2(0) + \mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(0) + \mathbf{P}^2(0)\mathbf{P}(1) \tag{B.10}$$

can be converted into a scalar probability by premultiplying by the initial state distribution and postmultiplying by the all-ones vector:

$$P(E) = \boldsymbol{\pi}\mathbf{P}(E)\mathbf{1}$$

The resulting probability remains quite manageable for the Gilbert channel:

$$P(E) = P(1-h)\frac{(3h^2-4h+1)p^2 - (6h^2-4h-2Ph+2P-2)p + 3h^2}{P+p}$$

An error event may also include the occurrence of more or less than a given number of errors in a block. Continuing with the previous example, suppose E represents the occurrence of two or more errors in a block of three symbols. This gives $E = \{(110) \cup (011) \cup (111)\}$, the probability of which is:

$$\mathbf{P}(E) = \mathbf{P}^2(1)\mathbf{P}(0) + \mathbf{P}(0)\mathbf{P}^2(1) + \mathbf{P}(1)\mathbf{P}(1)\mathbf{P}(1) \tag{B.11}$$

There are several algorithms that will search through all the possible combinations that make up an error event. For the case of scalar probabilities, one can use the binomial formula to compute such quantities:

$$P(n,m,p,q) = \binom{n}{m} p^{n-m} q^m$$

where n is the number of trials, m the number of errors, p the success probability and q the error probability.

The binomial formula does not work for matrix probabilities (where $p = \mathbf{P}(0)$ and $q = \mathbf{P}(1)$ because matrix multiplication does not necessarily commute. However, one can use a recursive form of the binomial formula [54]:

$$P(n,m,p,q) = P(n-1,m,p,q)\cdot p + P(n-1,m-1,p,q)\cdot q$$

by setting the halting conditions on the recursion to sequences consisting of powers of p ($m = 0$) and powers of q ($n = m$) alone, i.e.:

$$P(n, m, p, q) = \begin{cases} p^n & m = 0 \\ q^n & m = n \\ P(n-1, m, p, q) \cdot p + \\ \quad P(n-1, m-1, p, q) \cdot q & \text{otherwise} \end{cases}$$

This formulation still has the problem that it cycles through every possible combination of m errors in n trials. The number of permutations quickly becomes unmanageable as the number of trials and number of errors increases; there are, for example, 591,600,030 different ways that five errors can occur in a block of 150 symbols. Fortunately, there is a better way!

B.5 Fast Matrix Probability Calculations

A fast method for calculating matrix probabilities falls out from an investigation of the recursive nature of the binomial coefficients. It operates by using lower power coefficients to calculate those with higher powers. The formula for computing a general binomial coefficient bears a strong resemblance to the method for modular exponentiation used in public key cryptographic algorithms [55].

At its most basic form, one can compute $\mathbf{P}(2, m)$ from $\mathbf{P}(1, m)$ by treating the block of two symbols as two blocks of one symbol, i.e.:

$$\begin{aligned} \mathbf{P}(2,0) &= \mathbf{P}(1,0)\mathbf{P}(1,0) \\ \mathbf{P}(2,1) &= \mathbf{P}(1,0)\mathbf{P}(1,1) + \mathbf{P}(1,1)\mathbf{P}(1,0) \\ \mathbf{P}(2,2) &= \mathbf{P}(1,1)\mathbf{P}(1,1) \end{aligned}$$

One can then compute $\mathbf{P}(4, m)$ from $\mathbf{P}(2, m)$ in a similar manner:

$$\begin{aligned}
\mathbf{P}(4,0) &= \mathbf{P}(2,0)\mathbf{P}(2,0) \\
\mathbf{P}(4,1) &= \mathbf{P}(2,0)\mathbf{P}(2,1) + \mathbf{P}(2,1)\mathbf{P}(2,0) \\
\mathbf{P}(4,2) &= \mathbf{P}(2,0)\mathbf{P}(2,2) + \mathbf{P}(2,1)\mathbf{P}(2,1) + \mathbf{P}(2,2)\mathbf{P}(2,0) \\
\mathbf{P}(4,3) &= \mathbf{P}(2,1)\mathbf{P}(2,2) + \mathbf{P}(2,2)\mathbf{P}(2,1) \\
\mathbf{P}(4,4) &= \mathbf{P}(2,2)\mathbf{P}(2,2)
\end{aligned}$$

The progression follows quite naturally for all powers of two:

$$\mathbf{P}(2^n + 1, m) = \sum_{k+l=m} \mathbf{P}(2^n, k)\mathbf{P}(2^n, l)$$

The binomial coefficients for non-powers of two are computed in a fashion similar to the method used for the modular exponentiation of numbers in public key cryptographic algorithms. The block length n is first converted into its constituent powers of two. The matrix probabilities associated with these binary coefficients are then used to compute the desired $\mathbf{P}(n, m)$.

A simple example will serve to illustrate the process. Consider the computation of $\mathbf{P}(6, m)$. Since $6 = 2^2 + 2^1$, one can begin by calculating $\mathbf{P}(4, m)$ and $\mathbf{P}(2, m)$. These two terms are then combined in the usual fashion:

$$\begin{aligned}
\mathbf{P}(6,0) &= \mathbf{P}(4,0)\mathbf{P}(2,0) \\
\mathbf{P}(6,1) &= \mathbf{P}(4,0)\mathbf{P}(2,1) + \mathbf{P}(4,1)\mathbf{P}(2,0) \\
\mathbf{P}(6,2) &= \mathbf{P}(4,0)\mathbf{P}(4,2) + \mathbf{P}(4,1)\mathbf{P}(2,1) + \mathbf{P}(4,2)\mathbf{P}(2,0) \\
\mathbf{P}(6,3) &= \mathbf{P}(4,1)\mathbf{P}(2,2) + \mathbf{P}(4,2)\mathbf{P}(2,1) + \mathbf{P}(4,3)\mathbf{P}(2,0) \\
\mathbf{P}(6,4) &= \mathbf{P}(4,2)\mathbf{P}(2,2) + \mathbf{P}(4,3)\mathbf{P}(2,1) + \mathbf{P}(4,4)\mathbf{P}(2,0) \\
\mathbf{P}(6,5) &= \mathbf{P}(4,3)\mathbf{P}(2,2) + \mathbf{P}(4,4)\mathbf{P}(2,1) \\
\mathbf{P}(6,6) &= \mathbf{P}(4,4)\mathbf{P}(2,2)
\end{aligned}$$

Or, written in more general notation:

$$\mathbf{P}(n, m) = \sum_{k+l=m} \mathbf{P}(n-j, k)\mathbf{P}(j, l)$$

Computation of matrix probabilities for values of n having more factors are done recursively. For example, to calculate $\mathbf{P}(87, m)$, one begins by finding the powers of 2 that sum to 87. These are 2^6, 2^4, 2^2 and 2^0. One then calculates $\mathbf{P}(1, m)$, $\mathbf{P}(2, m)$, $\mathbf{P}(4, m) \ldots \mathbf{P}(64, m)$ and begins multiplying them together. First, $\mathbf{P}(1, m)$ and $\mathbf{P}(4, m)$ are combined to calculate $\mathbf{P}(5, m)$. $\mathbf{P}(5, m)$ is combined with $\mathbf{P}(16, m)$ to get $\mathbf{P}(21, m)$. Finally, $\mathbf{P}(21, m)$ and $\mathbf{P}(64, m)$ give $\mathbf{P}(85, m)$.

B.6 Matrix Generating Functions

One can write matrix probabilities in a more compact and accessible form through the use of matrix generating functions [54]. Matrix generating functions prove quite useful in calculating the performance of error control codes.

The matrix generating function is defined as the z-transform of $\mathbf{P}(\xi, x)$:

$$\mathbf{\Phi}(\xi, z) = \sum_x \mathbf{P}(\xi, x) z^x$$

where $\xi \in \mathbb{Z}+$ is an index generally taken to represent the length of the block of symbols and $\mathbf{P}(\xi, x)$ is the matrix probability of receiving x out of ξ symbols in error. n will take the place of ξ and m of x when describing block lengths and errors (where ξ and x are positive integers).

Consider the somewhat more ambitions example of a block of four bits for the purpose of explaining matrix generating functions. One can write the error probabilities by enumerating all possible combinations:

$$\begin{aligned}
\mathbf{P}(4,0) &= \mathbf{P}^4(0) \\
\mathbf{P}(4,1) &= \mathbf{P}^3(0)\mathbf{P}(1) + \mathbf{P}^2(0)\mathbf{P}(1)\mathbf{P}(0) + \mathbf{P}(0)\mathbf{P}(1)\mathbf{P}^2(0) + \mathbf{P}(1)\mathbf{P}^3(0) \\
\mathbf{P}(4,2) &= \mathbf{P}^2(0)\mathbf{P}^2(1) + \mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(0)\mathbf{P}(1) + \mathbf{P}(1)\mathbf{P}^2(0)\mathbf{P}(1) \\
&\qquad + \mathbf{P}(0)\mathbf{P}^2(1)\mathbf{P}(0) + \mathbf{P}(1)\mathbf{P}(0)\mathbf{P}(1)\mathbf{P}(0) + \mathbf{P}^2(1)\mathbf{P}^2(0) \\
\mathbf{P}(4,3) &= \mathbf{P}(0)\mathbf{P}^3(1) + \mathbf{P}(1)\mathbf{P}(0)\mathbf{P}^2(1) + \mathbf{P}^2(1)\mathbf{P}(0)\mathbf{P}(1) + \mathbf{P}^3(1)\mathbf{P}(0) \\
\mathbf{P}(4,4) &= \mathbf{P}^4(1)
\end{aligned} \tag{B.12}$$

The generating function follows from the expansion by associating $\mathbf{P}(4,0)$ with z^0, $\mathbf{P}(4,1)$ with z^1 and so on. This gives:

$$\mathbf{\Phi}(\xi, z) = \mathbf{P}(4,0) + \mathbf{P}(4,1)z + \mathbf{P}(4,2)z^2 + \mathbf{P}(4,3)z^3 + \mathbf{P}(4,4)z^4$$

Substituting the full matrix probabilities for each $\mathbf{P}_\xi(x)$ and factoring the resulting expression gives:

$$\mathbf{\Phi}(\xi, z) = (\mathbf{P}(0) + \mathbf{P}(1)z)^4 \tag{B.13}$$

This remarkably compact expression contains all of the information of (B.12). In fact, one can write the generating function for an arbitrary length binary block by substituting the length of that block for the 4 in (B.13):

$$\mathbf{\Phi}(n, z) = (\mathbf{P}(0) + \mathbf{P}(1)z)^n \tag{B.14}$$

See Appendix B.9 for a proof of (B.14).

The matrix probabilities are coefficients in the power series. Hence, one can extract a particular coefficient by repeatedly differentiating the series with respect to z and substituting $z = 0$:

$$\mathbf{P}(\xi, x) = \frac{1}{x!}\frac{d^x}{dz^x}\mathbf{\Phi}(\xi, z)\big|_{z=0} \tag{B.15}$$

The factor of $1/x!$ serves to cancel out the constants that accumulate from the differentiation process. Care must be taken in differentiating $\mathbf{\Phi}_\xi(z)$ since matrix multiplication does not necessarily commute, i.e., $\frac{d}{dz}(\mathbf{P}(0) + \mathbf{P}(1)z)^n$ does not necessarily equal $n\mathbf{P}(1)(\mathbf{P}(0) + \mathbf{P}(1)z)^{n-1}$. This process allows us to calculate error probabilities through repeated differentiation and works well for small values of x.

B.7 Fourier Transform Computations

An arbitrary discrete time series $\tilde{x}[n]$ with period N (i.e., $\tilde{x}[n] = \tilde{x}[n + aN]\ \forall\ a \in \mathbb{Z}$) has a discrete Fourier series (DFS) representation $\tilde{X}[k]$ comprised of N harmonically related complex exponentials together with their respective weights. These complex exponentials have frequencies equal to the integer multiples of the fundamental frequency $2\pi/N$. This fundamental exponential is often written as W_N:

$$W_N = e^{-j(2\pi/N)}$$

Hence, one can write sample n of exponential k as:

$$e_k[n] = e^{j(2\pi/N)kn} = W_N^{-kn}$$

The transformation from the time domain to the frequency domain is called analysis:

$$\tilde{X}[k] = \sum_{n=0}^{N-1} \tilde{x}[n] W_N^{kn} \tag{B.16}$$

and the transformation from the frequency domain to the time domain is called synthesis:

$$\tilde{x}[n] = \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[k] W_N^{-kn} \tag{B.17}$$

This analysis uses the discrete Fourier transform (DFT), which, while similar to the DFS, differs in that the discrete time sequence is taken to be identically zero outside the interval $[0, 1, \ldots, N]$ instead of repeating the values in that interval in a modular fashion. One takes one cycle of $\tilde{x}[n]$ to get $x[n]$:

$$X[k] = \begin{cases} \sum_{n=0}^{N-1} \tilde{x}[n] W_N^{kn} & 0 \leq k \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

and, by duality, one cycle of $\tilde{X}[k]$ to get $X[k]$:

$$x[n] = \begin{cases} \frac{1}{N} \sum_{k=0}^{N-1} \tilde{X}[n] W_N^{-kn} & 0 \leq n \leq N-1 \\ 0 & \text{otherwise} \end{cases}$$

There exists a straightforward relationship between the Fourier transform and z-transform. Given the z-transform, one can find the Fourier transform by evaluating the z-transform around the unit circle through the substitution of $e^{j\omega}$ for z (assuming that the Fourier transform exits), i.e.:

$$\begin{aligned}\tilde{X}[k] &= X(z)|_{z=e^{j(2\pi/N)k}} \\ &= X(e^{j(2\pi/N)k}) \\ &= X(W_N^{-k})\end{aligned}$$

This gives us analysis and synthesis equations similar to (B.16) and (B.17):

$$\text{analysis:} \quad X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{B.18}$$

$$\text{synthesis:} \quad x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] W_N^{-kn} \tag{B.19}$$

In order to sample the Fourier transform to find the DFT, one must take more than N equally spaced samples to ensure the uniqueness of the resulting time domain sequence, where N is the number of terms in the z-transform [56]. Consider, for example, a hypothetical z-transform $X(z)$ associated with a time domain sequence $x[n]$ of length N. One can recover $x[n]$ from $X(z)$ by evaluating $X(z)$ at N equally spaced points around the unit circle via synthesis equation (B.19):

$$\begin{aligned}x[n] &= \frac{1}{N} \sum_{k=0}^{N-1} \left(X(z) W_N^{-kn}|_{z=W_N^{-k}} \right) \\ &= \frac{1}{N} \sum_{k=0}^{N-1} X(W_N^{-k}) W_N^{-kn}\end{aligned}$$

Thus, to find $\mathbf{P}(n,m)$ via an inverse Fourier transform of generating function $\boldsymbol{\Phi}(n,z)$ calculate (keeping in mind that $\boldsymbol{\Phi}(n,z) = (\mathbf{P}(0) + \mathbf{P}(1))^n$ for the binomial case) [54]:

$$
\begin{aligned}
\mathbf{P}(n,m) &= \frac{1}{N}\sum_{k=0}^{N-1}\left(\mathbf{\Phi}(n,z)^n W_N^{-km}|_{z=W_N^{-k}}\right) \\
&= \frac{1}{N}\sum_{k=0}^{N-1}\mathbf{\Phi}(n,W_N^{-k})^n W_N^{-km} \\
&= \frac{1}{N}\sum_{k=0}^{N-1}(\mathbf{P}(0)+\mathbf{P}(1)W_N^{-k})^n W_N^{-km}
\end{aligned}
$$

This procedure was used to calculate all of the examples in this dissertation.

B.8 Symbol-Based Codes

Reed-Solomon codes are based on symbols where each symbol is comprised of several bits (assuming a Reed-Solomon code over GF(2^u)). An error in one or more bits of a symbol causes a symbol error. Hence, it is necessary to generate new definitions for $\mathbf{P}(0)$ and $\mathbf{P}(1)$ when analyzing symbol-based codes.

Consider the Reed-Solomon code over GF(2^u) where each symbol consists of u bits. Given BPSK transmission, each symbol requires u separate transmissions. The probabilities of a successfully received symbol ($\mathbf{P}_u(0)$) and an erroneous symbol ($\mathbf{P}_u(1)$) are:

$$
\begin{aligned}
\mathbf{P}_u(0) &= \mathbf{P}(0,n) \\
\mathbf{P}_u(1) &= \sum_{k=1}^{u}\mathbf{P}(k,u)
\end{aligned}
$$

The error calculations then proceed as usual using the new success and failure probabilities, i.e.:

$$
\mathbf{P}_u(n,m) = \frac{1}{N}\sum_{k=0}^{N-1}(\mathbf{P}_u(0)+\mathbf{P}_u(1)W_N^{-k})^n W_N^{-km}
$$

gives the probability of receiving m out of n symbols incorrectly.

B.9 Proof of (B.14)

Claim: $\mathbf{\Phi}(n,z) = (\mathbf{P}(0)+\mathbf{P}(1)z)^n$. This is clear by inspection for a one symbol long block ($n = 1$): the symbol may be received either successfully or unsuccessfully, the respective matrix probabilities of which are $\mathbf{P}(0)$ and $\mathbf{P}(1)$. The generating function for the one symbol block is therefore $\mathbf{P}(0) + \mathbf{P}(1)z$.

If true for n, one can see that the formula will also work for $n + 1$:

$$\mathbf{\Phi}(n, z) \cdot \mathbf{\Phi}(1, z) = (\mathbf{P}(0) + \mathbf{P}(1)z)^n(\mathbf{P}(0) + \mathbf{P}(1)z)$$

$$\mathbf{\Phi}(n + 1, z) = (\mathbf{P}(0) + \mathbf{P}(1)z)^{n+1}$$

Q.E.D.

APPENDIX C

MODEL BUILDING

This chapter explains the technique used to create the HMMs used in this dissertation.

C.1 Matrix Distributions

One can model the state holding times using a phase-type distribution $f(x) = \boldsymbol{\pi}\mathbf{A}^{x-1}\mathbf{b}$ where $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ are such that:

$$\mathbf{P} = \begin{bmatrix} \mathbf{A} & \mathbf{b} \\ \boldsymbol{\pi} & 0 \end{bmatrix} \tag{C.1}$$

is a stochastic matrix, i.e., the rows sum to one ($\mathbf{P1} = \mathbf{1}$). This fact is used later on to simplify the calculations required to fit a phase-type distribution to sample data.

The initial choices for $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ are unimportant in determining the final solution, but good picks will speed the convergence of the algorithm. Different initial values of these three matrices were evaluated over multiple runs of the algorithm to find a single set of starting values that offered rapid convergence for a wide variety of state holding times. For $\mathbf{A}$, this was ones along the diagonal and small random numbers (≤ 0.1) in the off diagonal positions. For $\boldsymbol{\pi}$, a one in the next to the last position and 0.001 in the remaining positions, i.e., $\boldsymbol{\pi} = [0.001, \cdots, 0.001, 1, 0.001]$. For $\mathbf{b}$, 0.5 in the last position and 0.001 in the remaining positions, i.e., $\mathbf{b} = [0.001, \cdots, 0.001, 0.5]^T$.

$\boldsymbol{\pi}$ is normalized by dividing each element by the sum of all of the elements. $\mathbf{A}$, and $\mathbf{b}$ are normalized together in a row-by-row fashion. That is, each row in $\mathbf{A}$ is summed together with the corresponding entry in $\mathbf{b}$, and the elements are divided by that sum. The resulting matrix and vectors satisfy equation C.1 and are used to calculate $\mathbf{g}$ (1xl), an estimate of the sample data (1xl).

C.2 Matrix Reestimation

Updating the estimates of A (nxn matrix), $\boldsymbol{\pi}$ (1xn vector) and $\mathbf{b}$ (nx1 vector) given sample data (1xl) [54] is accomplished as follows:

Calculate $\boldsymbol{\rho}(t)$ (1xl vector):

$$\rho_m(t) = \frac{f_m}{g_m(t)} \tag{C.2}$$

Calculate $\boldsymbol{\alpha}(t)$ (lxn matrix):

$$\boldsymbol{\alpha}_{1,:}(t) = \boldsymbol{\pi}(t-1) \quad \boldsymbol{\alpha}_{x,:}(t) = \boldsymbol{\alpha}_{x-1,:}(t)\mathbf{A}(t-1) \quad x = 2\ldots \tag{C.3}$$

Calculate $\boldsymbol{\beta}$ (nxl vector):

$$\boldsymbol{\beta}_{:,1}(t) = \mathbf{b} \quad \boldsymbol{\beta}_{:,x}(t) = \mathbf{A}(t-1)\boldsymbol{\beta}_{:,x-1}(t) \quad x = 2\ldots \tag{C.4}$$

Reestimate $\boldsymbol{\pi}$ (nx1 vector):

$$\pi_i(t) = \pi_i(t-1)\boldsymbol{\rho}(t)\boldsymbol{\beta}_{i,:}(t) \tag{C.5}$$

Reestimate $\mathbf{A}$ (1xn vector):

$$a_{i,j}(t) = a_{i,j}(t-1)\sum_{x=1}^{l}\rho_x(t)\sum_{y=1}^{x-1}\alpha_{y,i}(t)\beta_{j,x-y}(t) \tag{C.6}$$

Reestimate $\mathbf{b}$ (nx1 vector):

$$b_{i,j}(t) = b_{i,j}(t-1)\sum_{x=1}^{l}\rho_x(t)\alpha_{x,i}(t) \tag{C.7}$$

C.3 Matrix Renormalization

The reestimated versions of $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ are then renormalized using the process described earlier in the chapter. It is possible to keep $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ normalized throughout the reestimation process, but the reestimation equations to do so are considerably more computationally complex than the ones given above. It is more efficient to renormalize $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ as a group once they have all been reestimated. This method has the additional advantage that it is not subject to accumulating numerical inaccuracy, since the reestimation procedure may continue for several thousand iterations.

The reestimated versions of $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ are then used to calculate $\mathbf{g}$, after which the MSE between $\mathbf{g}$, and is calculated. If the MSE is sufficiently small, or the last round of reestimation did not significantly improve the MSE, the reestimation process terminates.

C.4 State Error Probability Calculation

Once $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ have been calculated, the individual state error probabilities are calculated with:

$$Pe = \int_{x_n}^{x_{n+1}} \frac{\gamma}{\sigma^2} \exp\left\{ -\frac{\gamma^2}{2\sigma^2} \right\} \cdot \frac{1}{2} \text{erfc}\left\{ \frac{\gamma}{\sqrt{E_b/N_0}} \right\} \tag{C.8}$$

where $[x_0, x_1, \cdots, x_l]$ are the SNR boundaries for each region.

C.5 Matrix Combination

The individual $\boldsymbol{\pi}$, $\mathbf{A}$, and $\mathbf{b}$ matrices for each state, along with the state error probabilities, must then be combined into a single matrix for error pattern calculations. Given a model with m states, let $\boldsymbol{\pi}_1$, $\mathbf{A}_1$, and $\mathbf{b}_1$ be the matrices for state one, $\boldsymbol{\pi}_2$, $\mathbf{A}_2$, and $\mathbf{b}_2$ the matrices for state two, and so on.

Furthermore, let $\mathbf{a}$ be the matrix of state transition probabilities where $\mathbf{a}(i, j)$ indicates the probability of transitioning from state i to state j when the transition occurs. The overall state transition matrix $\mathbf{A}$ is then calculated as follows:

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{a}_{1,2} \cdot \mathbf{b}_1 \cdot \boldsymbol{\pi}_2 & \dots & \mathbf{a}_{1,8} \cdot \mathbf{b}_1 \cdot \boldsymbol{\pi}_8 \\ \mathbf{a}_{2,1} \cdot \mathbf{b}_2 \cdot \boldsymbol{\pi}_1 & \mathbf{A}_2 & \dots & \mathbf{a}_{2,8} \cdot \mathbf{b}_2 \cdot \boldsymbol{\pi}_8 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{a}_{8,1} \cdot \mathbf{b}_2 \cdot \boldsymbol{\pi}_1 & \mathbf{a}_{8,2} \cdot \mathbf{b}_2 \cdot \boldsymbol{\pi}_8 & \dots & \mathbf{A}_8 \end{bmatrix} \tag{C.9}$$

$\mathbf{e}$ is calculated as:

$$\mathbf{e} = (Pe_1, Pe_1, \cdots, Pe_1, Pe_2, Pe_2, \cdots, Pe_2, \cdots, Pe_{l-1}, Pe_{l-1}, \cdots, Pe_{l-1}) \tag{C.10}$$

overall success and failure matrices **P0** and **P1** are calculated as:

$$\mathbf{P0} = \mathbf{A} \cdot \mathbf{I} \cdot (\mathbf{1} - \mathbf{e}) \tag{C.11}$$

$$\mathbf{P1} = \mathbf{A} \cdot \mathbf{I} \cdot \mathbf{e} \tag{C.12}$$

and overall initial state distribution matrix $\boldsymbol{\pi}$ is calculated as:

$$\boldsymbol{\pi} = \lim_{k \to \infty} \mathbf{A} \cdot \begin{bmatrix} 1 & 1 & \dots & 1 \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix} \tag{C.13}$$

REFERENCES

[1] W. Turin and R. van Nobelen, "Hidden markov modeling of flat fading channels," *IEEE Journal on Selected Areas in Communications*, vol. 16, pp. 1809–1817, December 1998.

[2] A. Campbell, G. Coulson, F. Garcia, D. Hutchinson, and H. Leopold, "Integrated quality of service for multimedia communications," *IEEE INFOCOM*, vol. 2, pp. 732–739, 1993.

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

[4] N. Shacham, "Dynamic parameter selection in packet radio transmissions," *IEEE MILCOM*, vol. 1, no. 5, pp. 28.4.1–28.4.3, 1986.

[5] N. Shacham, "Adaptive link level protocol for packet radio channels," *IEEE INFOCOM*, no. 4, pp. 626–535, 1986.

[6] M. Rice and S. Wicker, "Adaptive error control for slowly varying channels," *IEEE Transactions on Communications*, vol. 42, pp. 917–925, February/March/April 1994.

[7] S. Yajnik, J. Sienicki, and P. Agrawal, "Adaptive coding for packetized data in wireless networks," *IEEE Conference on Personal, Indoor and Mobile Radio Communications*, vol. 1, no. 6, pp. 338–342, 1995.

[8] C. Schuler, "Optimization and adaptation of error control algorithms for wireless ATM," *International Journal of Wireless Information Networks*, vol. 5, no. 2, pp. 61–73, 1998.

[9] J. Gomez, A. T. Campbell, and H. Horikawa, "A systems approach to prediction, compensation and adaptation in wireless networks," *First ACM/IEEE International Workshop on Wireless and Mobile Multimedia*, vol. 1, no. 1, pp. 92–100, 1998.

[10] M. Elaoud and P. Ramanathan, "Adaptive use of error-correcting codes for real-time communication in wireless networks," *IEEE INFOCOM*, vol. 2, no. 17, pp. 548–555, 1998.

[11] P. Lettieri and M. Srivastava, "Adaptive frame length control for improving wireless link throughput, range, and energy efficiency," *IEEE INFOCOM*, vol. 2, no. 17, pp. 564–571, 1998.

[12] R. Chen, K. C. Chua, B. T. Tan, and C. S. Ng, "Adaptive error coding using channel prediction," *Wireless Networks*, vol. 5, pp. 23–32, January 1999.

[13] C. Chien, M. Srivastava, R. Jain, P. Lettieri, V. Aggarwal, and R. Sternowski, "Adaptive radio for multimedia wireless links," *IEEE Journal on Selected Areas in Communication*, vol. 17, pp. 793–813, May 1999.

[14] I. F. Akyildiz, I. Joe, H. Driver, and Y. L. Ho, "A new adaptive FEC scheme for wireless ATM networks," *IEEE MILCOM*, vol. 1, no. 18, pp. 277–281, 1998.

[15] I. F. Akyildiz and I. Joe, "TCP performance improvement over wireless ATM networks through a new aal protocol," *IEEE GLOBECOM*, vol. 1, no. 56, pp. 508–512, 1998.

[16] I. F. Akyildiz and I. Joe, "A new ARQ protocol for wireless ATM networks," *IEEE International Conference on Communications*, vol. 2, no. 34, pp. 1109–1113, 1998.

[17] E. N. Gilbert, "Capacity of a burst-noise channel," *Bell Systems Technical Journal*, vol. 39, pp. 1253–1264, September 1960.

[18] B. D. Frichtman, "A binary channel characterization using particioned markov chains," *IEEE Transactions on Information Theory*, vol. 13, pp. 221–227, April 1967.

[19] W. Turin and M. M. Sondhi, "Modeling error sources in digital channels," *IEEE Journal in Selected Areas in Communications*, vol. 11, pp. 340–347, April 1993.

[20] F. Swarts and H. Ferriera, "Markov characterization of digital fading mobile vhf channels," *IEEE Transactions on Vehicular Technology*, vol. 43, pp. 977–985, November 1994.

[21] M. Zorzi, R. R. Rao, and L. B. Milstein, "On the accuracy of a first-order markov model for data transmission on fading channels," *IEEE International Conference on Universal Personal Communications*, pp. 211–215, November 1995.

[22] H. S. Wang and N. Moayeri, "Finite-state markov channel - a useful model for radio communications channels," *IEEE Transactions on Vehicular Technology*, vol. 44, pp. 163–171, February 1995.

[23] S. Sivaprakasam and K. S. Shanmugan, "An equivalent markov model for burst errors in digital channels," *IEEE Transactions on Communications*, vol. 43, pp. 1347–1354, February/March/April 1995.

[24] M. Sajadieh, F. R. Kschischang, and A. Leon-Garcia, "A block memory model for correlated rayleigh fading channels," *IEEE International Conference on Communications*, vol. 32, pp. 282–286, June 1996.

[25] F. Babich and G. Lombardi, "A markov model for the mobile propagation channel," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 63–73, January 2000.

[26] M. Zorzi, A. Chockalingam, and R. R. Rao, "Throughput analysis of tcp on channels with memory," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 1289–1300, July 2000.

[27] C. C. Tan and N. C. Beaulieu, "On first-order markov modeling for the rayleigh fading channel," *IEEE Transactions on Communications*, vol. 48, pp. 2032–2040, December 2000.

[28] COST 207 TD(86)51-REV3 (WG1), "Proposal on channel transfer functions to be used in GSM tests late 1986," Paris, September 1986.

[29] F. Gagnon and D. Haccoun, "Bounds on the error performance of coding for non-independent rician-fading channels," *IEEE Transactions in Communications*, vol. 40, pp. 351–360, February 1992.

[30] S. Wicker, *Error Control Systems for Digital Communication and Storage.* Englewood Cliffs, NJ: Prentice Hall, 1995.

[31] G. L. Stüber, *Principles of Mobile Communications.* Boston, MA: Kluwer Academic, 1996.

[32] W. Press, S. Teukolsky, W. Vetterling, and B. Flannery, *Numerical Recipes in C.* New York, NY: Cambridge University Press, 2 ed., 1994.

[33] R. H. Clarke, "A statistical theory of mobile radio reception," *Bell System Technical Journal*, vol. 32, pp. 957–1000, July 1968.

[34] W. C. Jakes, *Microwave Mobile Communications.* New York, NY: Wiley, 1974.

[35] P. Dent, G. Bottomley, and T. Croft, "Jakes fading model revisited," *Electronic Letters*, vol. 7, pp. 1162–1163, June 1993.

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

[37] J. Proakis, *Digital Communications.* McGraw Hill, 1995.

[38] E. Kreyszig, *Advanced Engineering Mathematics.* New York: Wiley, 5 ed., 1983.

[39] R. McEliece and L. Swanson, "On the decoder error probability for reed solomon codes," *IEEE Transactions on Information Theory*, vol. 5, pp. 701–703, September 1986.

[40] F. MacWilliams and N. Sloane, *The Theory of Error Control Codes.* Amsterdam: North-Holland, 1977.

[41] W. Turin, "Throughput analysis of the go-back-n protocol in fading radio channels," *IEEE Journal on Selected Areas in Communications*, vol. 17, pp. 881–887, May 1999.

[42] W. Turin and M. Zorzi, “Performance analysis of delay-constrained communications over diverse burst-error channels,” *IEEE Vehicular Technology Conference*, vol. 3, no. 50, pp. 1305–1309, 1999.

[43] K. Cornett and S. Wicker, “Bit error rate estimation techniques for digital land mobile radios,” *IEEE Vehicular Technology Conference*, no. 41, pp. 543–548, 1991.

[44] L. K. Rasmussen, M. J. Bartz, and S. B. Wicker, “A comparison of trellis coded and reed-solomon coded hybrid-ARQ protocols over slowly fading rayleigh channels,” *Wireless Personal Communications*, vol. 2, pp. 393–413, April 1996.

[45] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding.* New York, NY: McGraw Hill, 1979.

[46] D. Gorenstein and N. Zierler, “A class of error correcting codes in p,” *Journal of the Society of Industrial and Applied Mathematics*, vol. 9, pp. 207–241, June 1961.

[47] J. L. Massey, “Shift register synthesis and bch decoding,” *IEEE Transactions on Information Theory*, vol. 18, pp. 122–127, January 1969.

[48] E. R. Berlekamp, *Algebraic Coding Theory.* New York: McGraw Hill, 1968.

[49] R. T. Chien, “Cyclic decoding procedure for the bose-chaudhuri-hocquenhem codes,” *IEEE Transactions on Information Theory*, vol. 10, pp. 357–363, October 1964.

[50] G. D. Forney, “On decoding bch codes,” *IEEE Transactions on Information Theory*, vol. 11, pp. 549–557, October 1965.

[51] G. D’Aria, L. Stola, and V. Zingarelli, “Modeling and simulation of the propagation characteristics of the 900 MHz narrowband-TDMA CEPT/GSM mobile radio,” *IEEE Vehicular Technology Conference*, vol. 39, pp. 631–639, 1988.

[52] J. Cai and D. Goodman, “General packet radio service in gsm,” *IEEE Communications Magazine*, vol. 35, pp. 122–131, October 1997.

[53] W. Tranter and K. Kosbar, "Simulation of communication systems," *IEEE Communications Magazine*, vol. 32, pp. 26–35, July 1994.

[54] W. Turin, *Digital Transmission Systems: Performance and Modeling.* New York, NY: McGraw Hill, 1999.

[55] K. Rosen, *Elementary Number Theory and its Applications.* Reading, Mass: Addison Wesley, 1993.

[56] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing.* Englewood Cliffs, NJ: Prentice Hall, 1989.

[57] S. Aikawa, Y. Motoyama, and M. Umehira, "Error correction and error detection techniques for wireless ATM systems," *Wireless Networks*, vol. 3, pp. 285–290, September 1997.

[58] R. Blahut, *Theory and Practice of Error Control Codes.* Reading, Massachusetts: Addison Wesley, 1983.

[59] A. DeSimone and S. Nanda, "Wireless data: Systems, standards, services," *Wireless Networks*, vol. 1, pp. 241–253, March 1995.

[60] D. S. Eom, M. Sugano, M. Murata, and H. Miyahara, "A combination scheme of ARQ and FEC for multimedia wireless ATM networks," *IEICE Transactions on Communications*, vol. E81-B, pp. 1016–1024, May 1998.

[61] J. Hagenauer and P. Hoher, "A viterbi algorithm with soft-decision outputs and its applications," *IEEE GLOBECOM*, vol. 47, no. 3, pp. 1680–1686, 1989.

[62] P. Jung and P. Baier, "On the representation of cpm signals by linear superposition of impulses in the bandpass domain," *IEEE Journal of Selected Areas in Communications*, vol. 10, pp. 1236–1242, October 1992.

[63] H. Stark and J. W. Woods, *Probability, Random Processes, and Estimation Theory for Engineers.* Englewood Cliffs, NJ: Prentice Hall, 2nd ed., 1994.

[64] S. Tsai, "Markov characterization of the hf channel," *IEEE Transactions on Communication*, vol. 17, pp. 24–32, February 1969.

[65] H. Viswanathan, "Capacity of markov channels with receiver csi and delayed feedback," *IEEE Transactions on Information Theory*, vol. 45, pp. 761–787, March 1999.

[66] H. S. Wang and N. Moayeri, "Modeling, capacity, and joint source/channel coding for rayleigh fading channels," *IEEE Vehicular Technology Conference*, no. 43, pp. 473–479, 1993.

[67] J. R. Yee and E. J. Weldon, "Evaluation of the performance of error correcting codes on a gilbert channel," *IEEE Transactions on Communications*, vol. 43, pp. 2316–2323, August 1995.

[68] M. Zorzi, "Power control and diversity in mobile radio cellular systems in the presence of ricean fading and log-normal shadowing," *IEEE Transactions in Vehicular Technology*, vol. 45, pp. 373–382, May 1996.

[69] M. Zorzi and R. R. Rao, "Throughput of selective-repeat ARQ with time diversity in markov channels with unreliable feedback," *Wireless Networks*, vol. 2, pp. 63–75, January 1996.

[70] M. Zorzi and R. R. Rao, "ARQ error control for delay-constrained communications on short-range burst-error channels," *IEEE Vehicular Technology Conference*, vol. 3, pp. 1528–1532, December 1997.

[71] M. Zorzi, R. R. Rao, and L. B. Milstein, "ARQ error control for fading mobile radio channels," *IEEE Transactions on Vehicular Technology*, vol. 46, p. 1997, May 445-455.

[72] M. Zorzi and R. R. Rao, "Energy constrained error control for wireless channels," *IEEE Personal Communications Magazine*, vol. 4, pp. 27–33, December 1997.

[73] M. Zorzi and R. R. Rao, "On the statistics of block errors in bursty channels," *IEEE Transactions on Communications*, vol. 45, pp. 660–667, June 1997.

[74] M. Zorzi, R. R. Rao, and L. B. Milstein, "Error statistics in data transmissions over fading channels," *IEEE Transactions on Communications*, vol. 46, pp. 1468–1477, November 1998.

[75] M. Zorzi, "Performance of FEC and ARQ error control in bursty channels under delay constraints," *IEEE Vehicular Technology Conference*, vol. 2, pp. 1390–1394, 1998.

[76] M. Chiani and A. Volta, "Hybrid ARQ/FEC techniques for wireless ATM local area networks," *IEEE Conference on Personal, Indoor and Mobile Radio Communications*, vol. 3, no. 7, pp. 898–901, 1996.

[77] A. Goldsmith and S. Chua, "Adaptive coded modulation for fading channels," *IEEE Transactions on Communications*, vol. 46, pp. 595–602, May 1998.

[78] S. Wicker, "High-reliability data transfer over the land mobile radio channel using interleaved hybrid-ARQ error control," *IEEE Transactions on Vehicular Technology*, vol. 39, pp. 48–55, February 1990.

[79] C. Sinner and R. Sigle, "Toward wireless multimedia communications: Current standards and future directions," *International Journal of Wireless Information Networks*, vol. 5, pp. 61–73, January 1998.

[80] J. B. Cain and D. McGregor, "A recommended error control architecture for ATM networks with wireless links," *IEEE Journal on Selected Areas in Communication*, vol. 15, pp. 16–28, January 1997.

[81] S. B. Wicker, "Reed-solomon error control coding for rayleigh fading channels with feedback," *IEEE Transactions on Vehicular Technology*, vol. 41, pp. 124–133, May 1992.

[82] S. Pejhan, M. Schwartz, and D. Anastassiou, "Error control using retransmission schemes in multicast transport protocols for real-time media," *IEEE/ACM Transactions on Networking*, vol. 4, pp. 413–427, June 1996.

[83] Y. Y. Kim and S. Li, "Capturing important statistics of a fading/shadowing channel for network performance analysis," *IEEE Journal of Selected Areas in Communication*, vol. 17, pp. 888–901, May 1999.

[84] M. J. Chu and W. E. Stark, "Effect of mobile velocity on communications in fading channels," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 202–210, January 2000.

[85] P. Orten and A. Svensson, "Sequential decoding in future mobile communications," *IEEE Conference on Personal, Indoor and Mobile Radio Communications*, vol. 3, no. 8, pp. 1186–1190, 1997.

[86] F. Swarts and H. Ferreira, "Markov characterization of channels with soft decision outputs," *IEEE Transactions on Communications*, vol. 41, pp. 678–682, May 1993.

[87] A. Chockaligam, W. Xu, M. Zorzi, and L. B. Milstein, "Throughput-delay analysis of a multichannel wireless access protocol," *IEEE Transactions on Vehicular Technology*, vol. 49, pp. 661–671, March 2000.

[88] C. Anton-Haro, J. A. R. Fonollosa, and C. Fauli, "On the inclusion of channel's time dependence in a hidden markov model for blind channel estimation," *IEEE Transactions on Vehicular Technology*, vol. 50, pp. 867–873, May 2001.

[89] W. K. M. Ahmed and P. J. McLane, "Random coding error exponents for flat fading channels with realistic channel estimation," *IEEE Journal on Selected Areas in Communications*, vol. 18, pp. 369–379, March 2000.

[90] F. Babich, O. E. Kelly, and G. Lombardi, "Generalized markov modeling for flat fading," *IEEE Transactions on Communications*, vol. 48, pp. 547–551, April 2000.

[91] M. R. Hueda, "A markov-based model for performance evaluation in multimedia cdma wireless transmission," *IEEE Vehicular Technology Conference*, vol. 2, no. 52, pp. 668–673, 2000.

[92] D. Huang and J. J. Shi, “Tcp over packet radio link with adaptive channel coding,” *IEEE Vehicular Technology Conference*, vol. 2, no. 51, pp. 790–794, 2000.

[93] A. Adbi, “Comments on “on verifying the first-order markovian assumption for a rayleigh fading channel model”,” *IEEE Transactions on Vehicular Technology*, vol. 48, p. 1739, September 1999.

[94] Q. Zhang and S. A. Kassam, “Finite-state markov model for rayleigh fading channels,” *IEEE Transactions on Communications*, vol. 47, pp. 1688–1692, November 1999.

[95] Y. L. Guan and L. F. Turner, “Generalised fsmc model for radio channels with correlated fading,” *IEE Proceedings-Communications*, vol. 146, pp. 133–137, April 1999.

[96] S. Shamai and O. Somekh, “Shannon-theoretic approach to a gaussian cellular multiple-access channel with fading,” *IEEE Transactions on Information Theory*, vol. 46, pp. 1401–1425, July 2000.

VITA

Andreas Yankopolus was born on October, 23$^{\text{rd}}$ 1972 in Karlsruhe, West Germany. He received his B.Sc. in Electrical Engineering from Carnegie Mellon University in 1994 and completed his Ph.D. in Electrical and Computer Engineering from Georgia Institute of Technology in 2004.

He is currently employed as a Senior Engineer in the wireless and networking division of Scientific Research Corporation in Atlanta, GA. His research interests include mobile ad-hoc networking, networking and MAC layer protocol design, hardware in-the-loop simulation, and the modeling of digital communications systems.